UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Herramienta interactiva para la práctica musical

Adrián Cobo Hermoso
Tutor: Alberto Suárez González

Junio 2018

Herramienta interactiva para la práctica musical

AUTOR: Adrián Cobo Hermoso
TUTOR: Alberto Suárez González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2018

Resumen (castellano)

En este Trabajo de Fin de Grado se ha desarrollado una aplicación informática para la realización de ejercicios vocales y de teoría musical. A través de estos ejercicios los usuarios podrán adquirir conocimientos de solfeo y armonía. También le permitirán trabajar tanto el oído musical como la afinación y mejorar la calidad de la voz. Tras analizar diferentes aplicaciones de entrenamiento musical disponibles, se ha decidido construir una aplicación orientada a usuarios que no tengan un amplio conocimiento musical. Por ello, en el diseño se ha primado la accesibilidad y facilidad de uso, de forma que los usuarios puedan aprender de una manera sencilla pero completa.

Los ejercicios de entrenamiento vocal tienen como objetivo mejorar la afinación para diferentes notas e intervalos. Para capturar la voz, se utiliza el micrófono que generalmente están disponibles en ordenadores de sobremesa y portátiles. Para el análisis de frecuencias se han utilizado algoritmos de transformada de Fourier FFT (Fast Fourier Transform) y DCT (Discrete cosine transform). Estas herramientas permiten identificar las notas que el usuario está cantando y determinar si su afinación es correcta.

Los ejercicios de teoría musical están basados en el reconocimiento de notas, intervalos y acordes. En concreto, se han diseñado ejercicios en los que el usuario ha de reconocer notas sintetizadas y emitidas por los altavoces del ordenador. Asimismo, tendrá que reconocer o completar intervalos y acordes con las notas correspondientes. Todos estos ejercicios pueden ser realizados en tres niveles de dificultad. Esto permite la adaptación a usuarios con diferente nivel de formación musical. De esta forma se hace también posible el progreso a medida que se van superando los ejercicios. Se espera que los usuarios que completen la secuencia de ejercicios incluidos en la aplicación adquieran gracias a esta un buen dominio de las bases de la armonía musical.

Abstract (English)

In this Final Degree Project, a computer application has been developed for performing vocal exercises and music theory. Through these exercises, users can acquire knowledge of solfege and harmony. They will also allow you to work on both the musical ear and the tuning and improve the quality of the voice. After analyzing different available music training applications, it has been decided to build an application aimed at users who do not have a broad musical knowledge. Therefore, design has prioritized accessibility and ease of use, so that users can learn in a simple but complete way.

The vocal training exercises aim to improve the tuning for different notes and intervals. To capture the voice, the microphone that is usually available on desktops and laptops is used. For the analysis of frequencies, Fourier transform algorithms FFT (Fast Fourier Transform) and DCT (Discrete cosine transform) have been used. These tools allow to identify the notes that the user is singing and determine if their tuning is correct.

The musical theory exercises are based on the recognition of notes, intervals and chords. Specifically, exercises have been designed in which the user has to recognize notes synthesized and emitted by the computer speakers. Likewise, you will have to recognize or complete intervals and chords with the corresponding notes. All these exercises can be done in three levels of difficulty. This allows adaptation to users with various levels of musical training. In this way, progress is also made possible as the exercises are overcome. It is expected that users who complete the sequence of exercises included in the application acquire thanks to this a good command of the bases of musical harmony.

Palabras clave (castellano)

Audio, cantar, sonido, análisis, transformada, Fourier, notas, acordes, intervalos, ejercicios, aplicación.

Keywords (English)

Audio, singing, sound, analysis, transformed, Fourier, notes, chords, intervals, exercises, application.

Agradecimientos

A mis profesores de la escuela de música, a mis amigos y a mi familia.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

Se desconoce el inicio de la música, pero sabemos que hoy sigue presente. Aun cambiando los estilos musicales a lo largo de la historia pasando por varias etapas, se establecieron unas bases que hoy en día se siguen estudiando y practicando. La base fundamental de toda esta música es la conocida como armonía clásica. La armonía clásica podría definirse como el conjunto de métodos utilizados durante toda la música clásica para la creación de las obras. Actualmente existe otro tipo de armonía conocida como armonía moderna. Esta se centra en los estilos actuales como el jazz o el pop.

Esta armonía es muy importante para aquellas personas que quieran realizar estudios musicales con instrumentos de lo que llamamos música actual, como la guitarra, el teclado o el bajo eléctrico ya que el estudio de estos instrumentos conlleva también el estudio de la armonía moderna. Deberemos ser capaces de poder saber qué notas o acorde se está tocando, así como poder realizar diferentes enlaces entre ellos que nos permita componer una obra completa.

1.1 Motivación

Dado el gran impacto que tiene actualmente la armonía moderna dentro del ámbito musical, se ha decidido realizar una aplicación que nos permita poder obtener una buena base para los diversos temas que abarca la armonía moderna. La gran mayoría de las aplicaciones actuales se centran en dar una gran abundancia de datos y tipos de ejercicios de manera que es más difícil para aquellos usuarios que no tienen todos esos conocimientos poder realizarlos. Esta aplicación se centra en poder dar un primer acercamiento menos frustrante para los usuarios que tienen interés por las bases de la armonía moderna. La incorporación además de ejercicios vocales dentro de la aplicación podrá ayudarnos a realizar de una manera práctica los ejercicios teóricos que se van aprendiendo.

1.2 Objetivos

Los objetivos se pueden detallar como los siguientes:

- Estudiar el comportamiento de señales de audio digitales y analógicas.
- Conocer y aprender a manejar algoritmos de uso común para señales (FFT y DCT).
- Estudio y uso de la librería JavaFX como interfaz gráfica.
- Uso de APIs externas para facilitar la tarea de programación base de la aplicación como JFugue, JTransforms y abc4j.
- Investigación de los ejercicios más simples y comunes sobre bases musicales como identificación de notas, intervalos y acordes.

Existen muchas aplicaciones hoy en día para cantar y acompañar a tus grupos favoritos, pero una herramienta más teórica nos permitiría mejorar el conocimiento adquirido que podríamos aplicar a esas otras aplicaciones cantando.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Un primer capítulo que muestre el estado actual de las aplicaciones relacionadas con el entrenamiento vocal y la música.
- Un segundo capítulo recapitulando los requisitos básicos de la aplicación que queremos desarrollar.

- Un tercer capítulo de investigación sobre las propiedades del sonido y cómo utilizarlas para un análisis.
- Un cuarto capítulo que explique los algoritmos utilizados para el análisis.
- Un quinto capítulo detallando las bibliotecas y herramientas software utilizadas para la codificación de la aplicación.
- Un sexto capítulo explicando cómo se ha desarrollado la aplicación.
- Un séptimo capítulo que muestra la estructura interna del programa y la organización de las clases Java utilizadas.
- Un último capítulo que muestra la aplicación final para el usuario y su interfaz gráfica

2 Estado del arte

Actualmente existen diversas aplicaciones que nos pueden ayudar tanto a nivel técnico musical como a nivel de usuario a cantar. El primer ejemplo que podríamos analizar sería el videojuego "SingStar [1]" para diferentes plataformas (PS2, PS3, PS4, Xbox, Xbox 360, Xbox One…), o la versión gratuita para PC "Ultrastar [2]". Estos videojuegos nos permiten cantar junto a nuestro artista favorito canciones a modo de karaoke, pero con la diferencia de que el videojuego analiza la voz para poder así saber si estamos cantando bien o mal la canción en función de nuestra afinación, es decir, analiza las notas que estamos cantando y con un margen de error nos puntúa si es correcta o no según avanza la letra. Existen diferentes niveles de dificultad que reducen ese margen de error para así obligar al usuario a obtener una afinación perfecta si quiere conseguir la máxima puntuación.

Dentro de los ejemplos de aplicaciones teóricas, encontramos diferentes páginas web y aplicaciones para dispositivos móviles que realizan ejercicios teóricos sobre todo tipo de conceptos, desde reconocer notas, hasta diferentes tonalidades y escalas. Una de las aplicaciones que se ha analizado ha sido la aplicación web "Music Theory [3]" la cual también está disponible para dispositivos móviles. Uno de los fallos que podríamos comentar es la curva de dificultad de aprendizaje a través de esta aplicación la cual es demasiado elevada desde un principio. Todos los ejercicios tienen un solo nivel de dificultad lo cual no facilita nada a usuarios con poco conocimiento. Aunque existe un apartado para poder configurar los ejercicios, requieren amplios conocimientos para su uso ya que realmente está diseñado para que un profesor pueda proponer ejercicios a sus alumnos.

Este es el problema general de todas las aplicaciones que encontramos actualmente en el mercado. La mayoría tiene un acceso difícil dada su curva de dificultad. Otro de los problemas que se encuentra actualmente es que la mayoría de estas aplicaciones, o al menos las más completas, requieren suscripciones y pagos para poder desbloquear ejercicios completos lo que hace que el usuario prefiera evitar este tipo de aplicaciones.

Cuando buscamos una herramienta que además unifique ejercicios teóricos y ejercicios vocales y de afinación la situación cambia un poco, pero esta vez a peor. Actualmente no existen aplicaciones que unifiquen ambos conceptos.

3 Análisis de requisitos

Tras haber analizado las diferentes soluciones actuales parecidas a la herramienta que queremos desarrollar, podemos listar los requisitos que debemos cumplir con nuestra aplicación. De una manera general podremos resumirla en los siguientes:

- La aplicación debe ser capaz de analizar el audio cantado por el usuario y dar una respuesta ante la afinación de la nota propuesta
- Debemos dividir en dos grandes bloques los ejercicios: vocales y teóricos.
- La aplicación debe contar con ejercicios que vayan partiendo de conocimientos adquiridos en ejercicios anteriores de manera que podamos aplicar en ejercicios más avanzados lo aprendido en los ejercicios anteriores.
- Se deben generar ejercicios de manera aleatoria y deben tener diferentes niveles de dificultad para los usuarios.

Estos requisitos son los principales que debe cumplir la aplicación. Así mismo se ha realizado una lista de los ejercicios de cada tipo que deberemos implementar en nuestra aplicación. En los ejercicios vocales nos centraremos en 4 tipos:

- Afinación de una nota concreta
- Afinación de una nota concreta con una referencia de un intervalo
- Afinación de un intervalo completo cantando una nota tras otra.
- Ejercicio de potencia, duración y respiración de voz

Para los ejercicios teóricos deberemos centrarnos en tres aspectos diferentes, las notas, los intervalos y los acordes por lo que realizaremos 5 ejercicios diferentes:

- Reconocer una nota (oído absoluto)
- Reconocer un intervalo
- Completar un intervalo
- Reconocer un acorde
- Completar un acorde

Observando los diferentes requisitos y analizando diferentes lenguajes de programación al final se ha decidido utilizar un lenguaje orientado a objetos, en concreto Java. Esto nos permitirá realizar buenos diseños de clases para los ejercicios ahorrándonos mucho trabajo en futuras iteraciones. Además, al ser un lenguaje tan extendido, existen numerosas librerías que nos van a poder ayudar a realizar la aplicación. Para poder desarrollar cómodamente la aplicación se van a realizar 5 iteraciones importantes:

- Diseño de clases orientadas a sonido
- Diseño de clases de ejercicios vocales
- Diseño de clases de ejercicios teóricos
- Diseño de clases para la interfaz gráfica
- Interfaz gráfica y configuraciones

Estas 5 iteraciones nos van a permitir trabajar de una manera ordenada y eficiente. Con ayuda de la aplicación Trello, podemos organizar las tareas que deberemos realizar en nuestro proyecto y poder llevar un control de cómo avanza el proyecto. Con el sistema de comentarios y anotaciones además podemos comentar problemas surgidos durante la implementación.

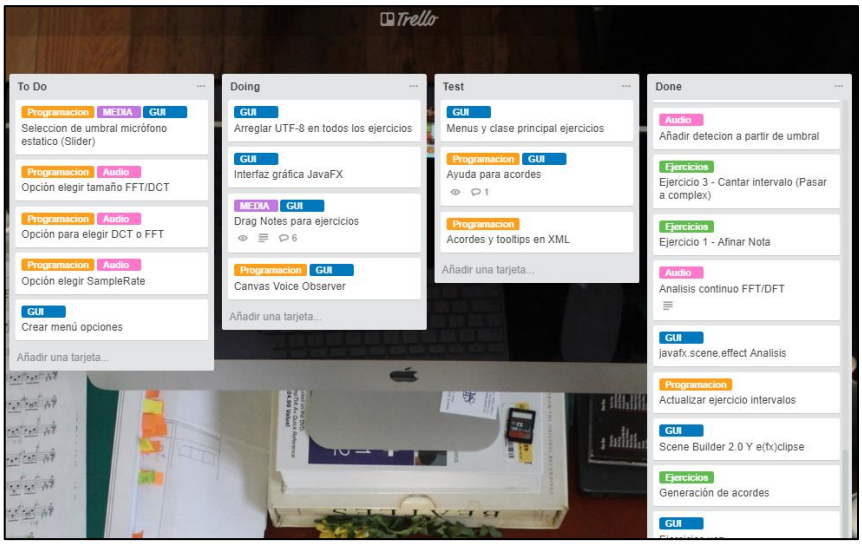


Figura 3-1: Trello y tareas asignadas

Aparte de Trello, se ha utilizado un control de versiones como es BitBucket que nos permite observar los cambios realizados en el código y tener un control total sobre las versiones que van surgiendo a lo largo del desarrollo. Esto nos permite mantener ordenado el código sabiendo exactamente, mediante comentarios, qué se ha añadido, modificado o eliminado dentro del código.

Figura 3-2: BitBucket

Tras haber analizado y listado los diferentes requisitos que deberemos cumplir en nuestra aplicación, comenzaremos a realizar un estudio sobre el sonido para poder entender las características y luego poder analizarlo dentro de nuestra aplicación y aplicarlo a nuestros ejercicios vocales.

4 El sonido

En este apartado comenzaremos un estudio previo del sonido. En primer lugar, definiremos las tres propiedades más importantes del sonido. Estas propiedades nos permitirán saber mejor cómo funcionan las ondas de sonido que nos interesan analizar en el proyecto. Una vez definidas, avanzaremos a un segundo capítulo en que veremos cómo estas propiedades se dan en la voz humana.

4.1 Propiedades del sonido

Cuando queremos categorizar el sonido hablamos de tres propiedades importantes. Intensidad, frecuencia y timbre. Cada una de estas propiedades nos permite distinguir un sonido de una fuente u otra. Por ejemplo, gracias a esto podemos distinguir una tromba de una flauta o de un piano. Aunque todos estos instrumentos toquen la misma nota, el sonido que provocarán será diferente en cada uno de los casos. En este capítulo haremos una breve introducción de las tres características del sonido.

4.1.1 Intensidad

La intensidad también es conocida como la amplitud. Esta propiedad es la que define la fuerza del sonido, es decir, el volumen con el que la señal es emitida desde su fuente. Esta amplitud es la que define la potencia de una señal de audio. La intensidad suele medirse en decibelios ya que nos da una medida más visual que una medida lineal. En general se toma como 0dB el nivel más bajo de audición y aumenta a partir de ahí.

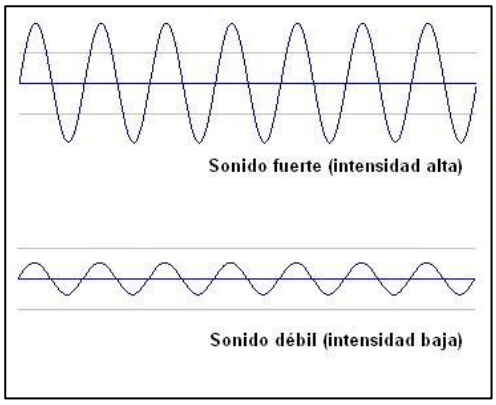


Figura 4-1: Intensidad [6]

4.1.2 Frecuencia

Esta propiedad es la que nos permite obtener el tono del sonido. Esta propiedad es medida en Hz y es la velocidad a la que vibra el sonido. Si vibra lentamente, la frecuencia generada será baja y provocará sonidos graves mientras que, si vibra rápidamente, la frecuencia que generará entonces será una frecuencia aguda. El humano tiene un umbral de frecuencias de entre 20 Hz y 20kHz.

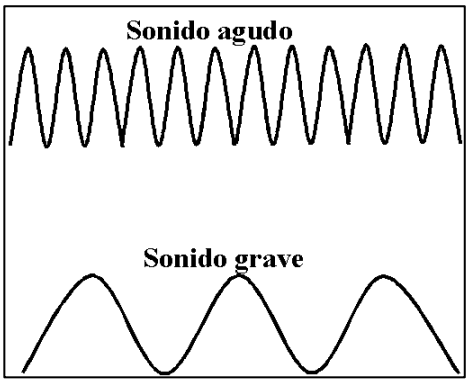


Figura 4-2: Frecuencia [6]

4.1.3 Timbre

El timbre no tiene una unidad de medida pues es una combinación de las dos propiedades anteriores.

Es la propiedad que nos permite distinguir fácilmente dos sonidos de fuentes diferentes. Sin esta última propiedad no podríamos distinguir entre una trompeta y un clarinete o una guitarra y no tendrían sentido esas grandes orquestas de numerosos instrumentos.

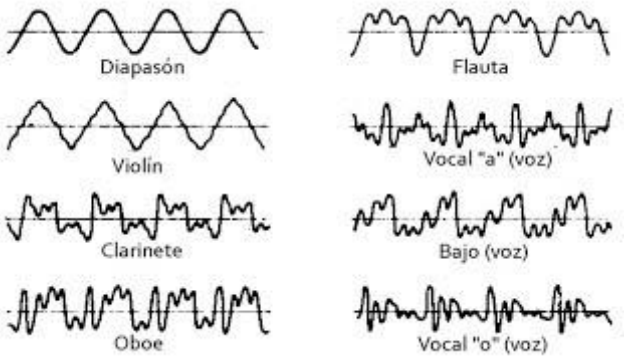


Figura 4-3: Timbre [4]

4.2 Propiedades de la voz humana

Tras analizar las propiedades generales del sonido debemos centrarnos en lo que nos interesa en este proyecto, la voz humana. La intensidad se localiza entre los 20dB y los 90 dB como máximo. Como esto son valores actualmente abstractos para nosotros comparemos estas intensidades con otras más conocidas. En los 20dB se localizaría aproximadamente el nivel de intensidad que se encuentra en las bibliotecas actualmente y en los 90dB sería el nivel que genera el tráfico en las ciudades. Por debajo de esos 20dB encontraríamos a 10dB el nivel de ruido que encontramos en el campo y por encima de los 90dB encontraríamos a 100dB el ruido que provoca una perforadora eléctrica.

Cuando hablamos de frecuencias la voz humana se encuentra entre los 250Hz y 3kHz, aunque ciertos fonemas usados pueden llegar hasta los 8kHz. Esto es importante ya que no hace falta que analicemos los 20kHz máximos de la audición humana ya que supone un coste adicional el cual, como vemos, no es necesario.

Para hablar del timbre en la voz humana anteriormente hemos mencionado que el timbre no tiene una medida ya que es la combinación de las propiedades intensidad y frecuencia y vamos a verlo ahora con más profundidad. Cuando una persona canta una nota no simplemente emite una frecuencia a una intensidad determinada sino varias frecuencias y cada una de ellas a una intensidad. La suma de todas esas frecuencias e intensidades es lo que definen el timbre. Estas frecuencias e intensidades no son arbitrarias y se generan de una manera muy específica y es lo que conocemos como armónicos.

4.3 Armónicos

Los armónicos son las frecuencias adicionales que se generan a partir de la frecuencia fundamental. La frecuencia fundamental es la frecuencia más grave generada, es decir, la frecuencia plana de la nota que nosotros queremos cantar. A continuación, multiplicamos esa frecuencia por dos para obtener el primer armónico, por tres para el segundo armónico y así hasta alcanzar el número de armónicos deseado.

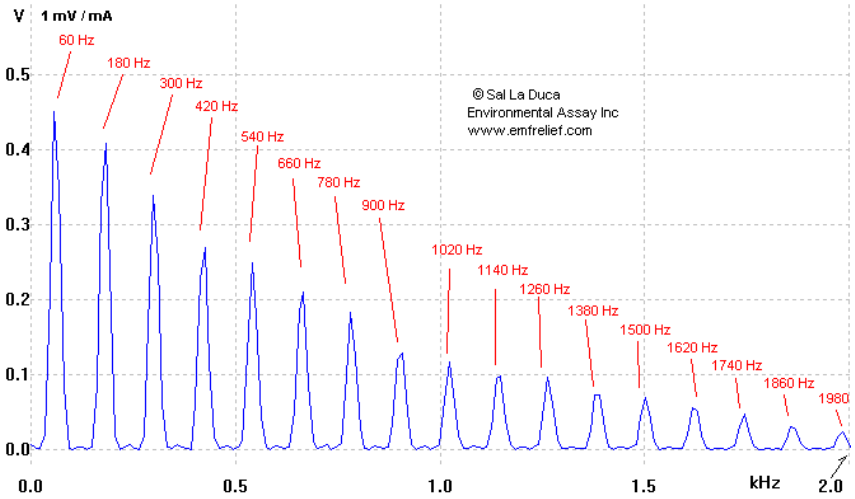


Figura 4-4: Representación de generación de armónicos [5]

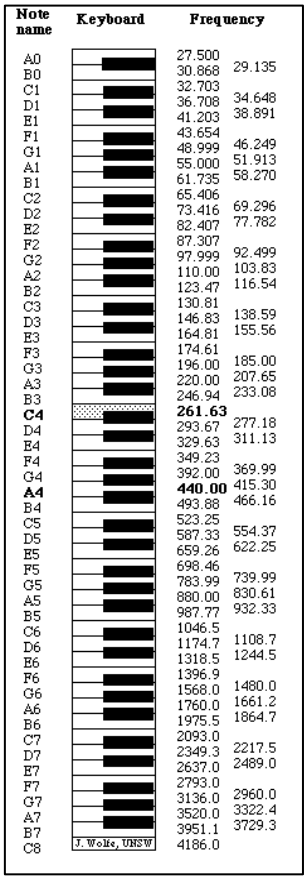


Figura 4-5: Frecuencias en el piano [7]

Cada nota tiene una correspondencia directa a una frecuencia fundamental. Esta correspondencia puede cambiar dependiendo el tipo de afinación que nosotros queramos aplicar. Para ello hablamos del A4 440Hz. El A4 es el concepto base para la afinación de cualquier instrumento y representa la nota La *(A es la notación americana)* de la cuarta octava del piano de rango completo.

Un piano puede tener escalas reducidas por lo que es importante hablar de un piano de rango completo. En este caso, el piano comienza siempre por la nota La *(A)* la cual será la octava cero. A continuación, se irán aumentando las escalas hasta que lleguemos a la nota La de la cuarta octava. Esta nota es A4 y deberá tener una frecuencia asociada. La frecuencia más usada es la 440Hz y 442Hz ya que son las usadas por instrumentos de cuerda y de viento clásicos, aunque cualquier afinador que nos permita cambiar esa frecuencia, permite oscilar entre 410Hz y 480Hz.

Para obtener la frecuencia que corresponde con cada nota de cada octava no es necesario tener una lista de frecuencias para cada nota y cada afinación de A4, podemos usar la siguiente fórmula. Siendo n la nota y o la octava:

$$f(n,o) = 440 * e^{\left((o-4)+\frac{(n-10)}{12}\right)*\ln(2)}$$

La nota será una de las 12 notas que vienen en este orden: C, C#, D, D#, E, F, F#, G, G#, A, A#, B siendo C=1 y B=12.

5 Herramientas para el análisis de señales acústicas

En este apartado analizaremos las soluciones actuales para poder realizar análisis de las señales acústicas para poder obtener la nota que el usuario está cantando. En primer lugar, veremos dos algoritmos que descomponen una señal en las diversas frecuencias encontradas y después explicaremos lo que son las ventanas de muestreo las cuales nos ayudan a poder mejorar el análisis de frecuencias.

5.1 Análisis de frecuencias

Cuando queremos saber qué nota se está emitiendo debemos fijarnos en su frecuencia fundamental. Esta frecuencia nos dirá exactamente qué nota está cantando el usuario. Para analizar el audio existen varios algoritmos de análisis que nos permiten obtener la descomposición en frecuencias de una onda de sonido. Según el uso que queramos aplicar a la descomposición deberemos elegir un tipo de algoritmo u otro. Aquí analizaremos los dos algoritmos más usados para estos trabajos.

5.1.1 FFT

El algoritmo más extendido para el análisis en frecuencia de audio es el FFT, Fast Fourier Transform o Transformada Rápida de Fourier [6]. Gracias a este eficiente algoritmo, podemos obtener la transformada de Fourier discreta y su inversa. La transformada de Fourier discreta nos permite obtener una descomposición en frecuencias de una onda que tiene que cumplir dos características importantes: ser discreta y continua. Además, para el análisis se presupone que la onda es periódica infinita. Como en el análisis de audio esto no se puede garantizar ya que se realizan numerosos muestreos por segundo, se aplican funciones ventana para reducir los espurios del espectro.

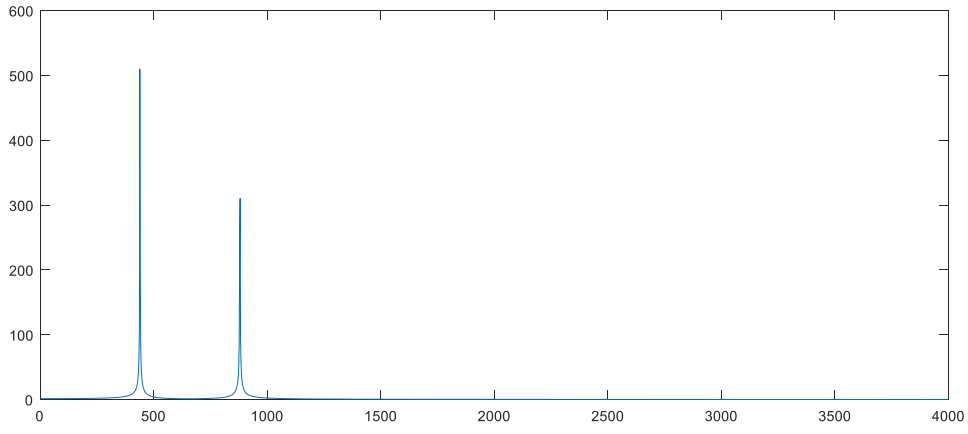


Figura 5-1: Ejemplo de algoritmo FFT

Para la ejecución del algoritmo mostrado, se ha usado un archivo de audio con una frecuencia de muestreo de 8000 Hz que contiene además dos ondas de 440Hz y 880Hz

La definición de la ecuación del algoritmo es la siguiente:

$$\{X_n\}\{x_n\} \subset \mathbb{C}$$

$$X_k = \sum_{n=0}^{N-1} \left(x_n e^{-\frac{2\pi i}{N} kn} \right) \quad k = 0, \dots, N-1$$

Como para este tipo de análisis se trabaja con números complejos y nuestra señal de entrada no es compleja, se utiliza como parte imaginaria i=0. De esta manera se puede realizar un análisis correcto y eficiente.

La ejecución completa para DFT requiere un total de N^2 operaciones mientras que su versión FFT se obtiene el mismo resultado con sólo $N * \log N$ operaciones. Para explicar y ver con mejor claridad el algoritmo FFT denotaremos la siguiente nueva variable.

$$W = e^{-\frac{2\pi i}{N} kn}$$

De esta manera la ecuación original resulta como:

$$X_k = \sum_{n=0}^{N-1} (x_n W)$$

Esta nueva W se puede calcular antes de la ejecución del algoritmo y luego usar los valores para la ejecución del algoritmo completa ahorrándonos mucho tiempo de cómputo.

El algoritmo más famoso para el cálculo de una FFT es el diseñado por J.W. Cooley y John Tukey en 1965 [7]. Es algoritmo es el conocido como "Algoritmo de diezmado en el tiempo". Este algoritmo divide la señal en dos muestras de N/2 con coeficientes pares e impares separados. Después los coeficientes impares son multiplicados por WNK (K = posición en el vector de salida) y a continuación se suman a las muestras pares.

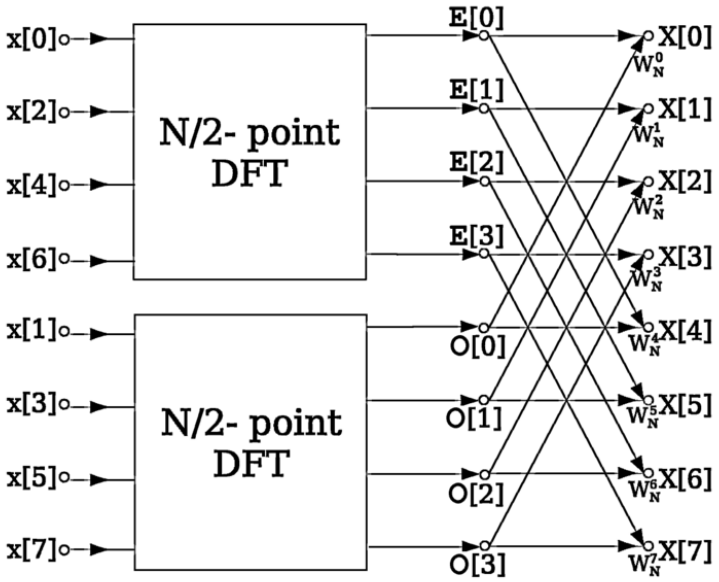


Figura 5-2: Ejemplo visual de FFT [7]

5.1.2 DST

El algoritmo DST (*Discrete Sine Transform*) [8] es un algoritmo parecido al de DFT, pero este solo trabaja con valores reales y además sólo única y exclusivamente con senos. Existe una variante de este algoritmo que solo trabaja con cosenos que sería la DCT (*Discrete Cosine Transform*) [9]. Tanto DCT como DST son funciones $\mathbb{R}^N \longrightarrow \mathbb{R}^N$ por lo que se puede representar también como una matriz de NxN. Una de las utilidades de DST y DCT es la compresión de datos, aunque no es muy aconsejable dada su tasa de pérdida se sigue utilizando hoy en día en algoritmos como:

- **DV (*Digital Video*):** formato estándar de video de gama doméstica, industrial y para transmisión.
- **AC-3:** compresión de audio que contiene hasta un total de 6 canales de audio con 5 canales de ancho de banda completa de 20Hz a 24kHz.
- **JPEG (*Joint Photographic Experts Group*):** es un estándar de compresión de imágenes fijas (fotografía).
- **MJPEG (*Motion JPEG*):** es usado como compresión de video en que cada fotograma es comprimido a JPEG.
- **MPEG-1, MPEG-2, MPEG-4:** estándares de codificación de audio y video. (https://commons.wikimedia.org/wiki/File:MPEG.svg)
- **Vorbis:** códec de audio que utiliza el contenedor Ogg (formato libre y abierto diseñado para proporcionar difusión de flujo eficiente y de alta calidad). Este usa también una versión modificada del DCT (MDCT) que mejora la cuantificación.

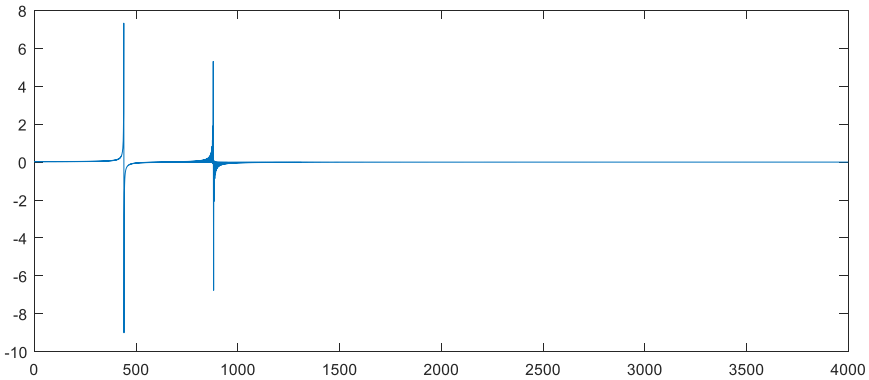


Figura 5-3: Ejemplo de algoritmo DCT onda de 440Hz y 880Hz

Tanto para DST como para DCT existen cuatro versiones fundamentales. Para las cuatro versiones, las matrices obtenidas son ortogonales, lo que quiere decir que la matriz inversa es igual a la matriz traspuesta:

- DST-I: Esta versión obtiene una matriz ortogonal, es decir, su matriz inversa es igual a la matriz traspuesta. (*Ej: DST(a,b,c) = DFT(0,a,b,c,0,-c,-b,-a) *½.*)

$$X_k = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N+1}(n+1)(k+1)\right]$$

- DST-II:

$$X_k = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N}\left(x+\frac{1}{2}\right)(k+1)\right]$$

- DST-III:

$$X_k = \frac{(-1)^k}{2} x_{n-1} + \sum_{n=0}^{N-2} x_n \sin\left[\frac{\pi}{N}(x+1)\left(k+\frac{1}{2}\right)\right]$$

- DST-IV:

$$X_k = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N}\left(x+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right]$$

- DCT-I:

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos\left[\frac{\pi}{N-1} nk\right]$$

- DCT-II:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right]$$

- DCT-III:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N} n\left(k+\frac{1}{2}\right)\right]$$

- DCT-IV:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right]$$

Además de las fórmulas a aplicar por el algoritmo, debemos seguir las condiciones de frontera asociadas a cada una de las versiones del algoritmo. En estas imágenes observamos las condiciones para DST y DCT.

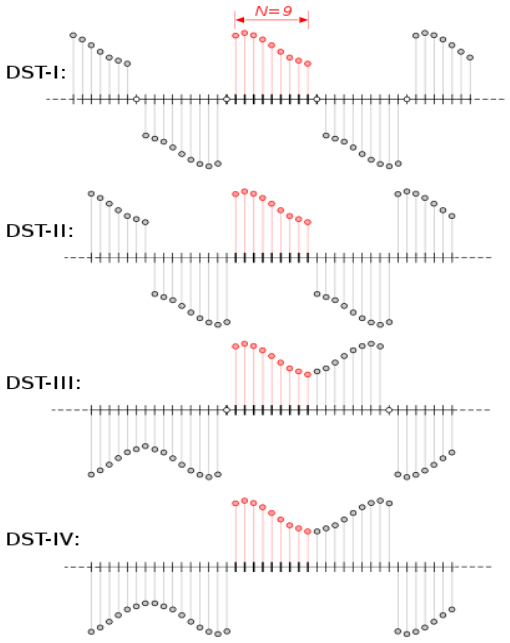


Figura 5-4: Condiciones de frontera DST [8]

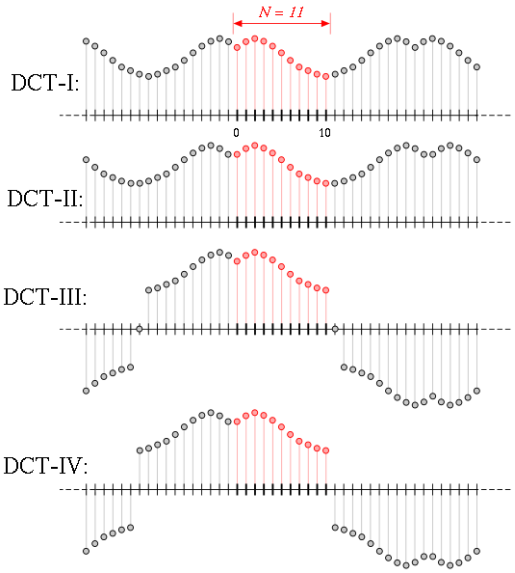


Figura 5-5: Condiciones de frontera DCT [9]

5.2 Ventanas de muestreo

Dado que el análisis de audio no es perfecto ya que realizamos muestreos continuos y las señales no son periódicas perfectas, para reducir espurios en el espectro aplicamos lo que se llaman "ventanas" que son funciones matemáticas destinadas a esto. Dependiendo de la función ventana que decidamos usar obtendremos diferentes resultados en el dominio de las frecuencias. Unas ventanas nos reducirán el dominio frecuencial del espectro mientras que otras lo ampliarán.

Para el análisis de frecuencias muy exactas necesitamos reducir mucho el dominio frecuencial para que nos dé exactamente cuál es la frecuencia que más represente nuestra onda de audio que hemos capturado. Por este motivo existen dos que son más usadas, la ventana de Hann y la ventana Hamming.

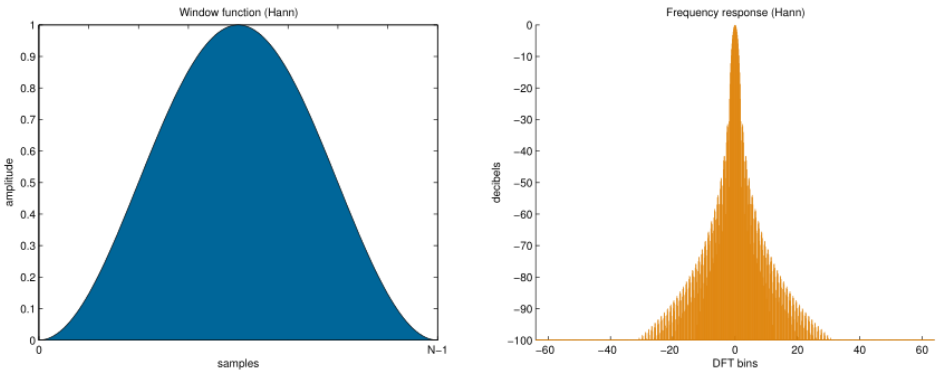


Figura 5-6:Ventana Hann y respuesta en frecuencia [10]

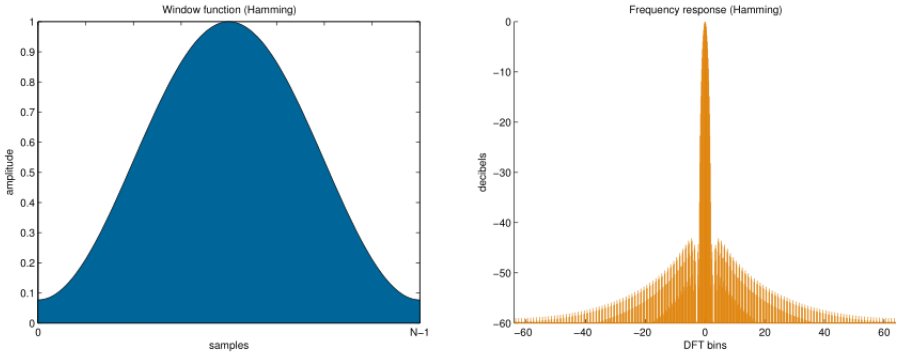


Figura 5-7: Ventana Hamming y repuesta en frecuencia [10]

Las ecuaciones de las ventanas son las mismas, pero utilizan diferentes constantes para modificar el espectro de una manera u otra.

$$v(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right)$$

Para la función de Hann usaremos $a_0 = a_1 = 0.5$ mientras que para la función Hamming usaremos $a_0 = 0.53836$ y $a_1 = 0.46164$. Gracias a esta función mejoraremos la frecuencia obtenida tras realizar un análisis de frecuencias.

6 Herramientas software

Tras haber comentado los diversos aspectos técnicos que nuestra aplicación debe cumplir, necesitamos implementarlos para poder seguir avanzando hacia nuestro objetivo. Dada la amplia variedad de librerías ya implementadas en Java, deberemos elegir las fundamentales que nos ayuden con nuestra tarea. Deberemos elegir una librería que nos permita realizar un análisis de las frecuencias de una muestra de audio, una librería que nos ayude con la reproducción de notas musicales, una librería que nos permita dar una interfaz gráfica que se ajuste a nuestras necesidades y por último una librería que nos permita mostrar visualmente las notas dentro de un pentagrama para una interfaz más atractiva al usuario.

6.1 JTransforms

Para la primera parte del trabajo, el análisis de audio en frecuencias, como ya hemos mencionado las mejores alternativas para analizar son FFT y DCT. Para ello lo mejor es utilizar la librería JTransforms "open source" que dispone de algoritmos de análisis de señales y además es multi-hilo. Esto nos permite mejorar la velocidad de procesado de los algoritmos ya que pueden ser implementados en paralelo totalmente.

Esta librería podemos usarla de una manera muy simple pasando el sample obtenido del micrófono y obteniendo los puntos para cada frecuencia. Esta herramienta no proporciona directamente la opción de obtener a qué frecuencia se refiere cada punto, pero no es necesario ya que solo necesitamos saber cuántos puntos hemos analizado para una frecuencia de muestreo concreta. De esta manera obtenemos un resultado rápido y sencillo que nos agiliza el proceso de análisis frecuencial.

6.2 JFugue

Para reproducir sonidos de instrumentos de una manera virtual se diseñó hace tiempo un protocolo llamado *MIDI* que nos permite poder reproducir notas de instrumentos sin tener que guardar las grabaciones de cada una de las obras. Cuando tenemos un audio de una canción, no se pueden modificar los instrumentos que suenan a lo largo de la canción. *MIDI* guarda las notas de cada uno de estos instrumentos y luego virtualiza los instrumentos de manera que podemos escuchar, cambiar instrumentos, notas, intensidades, tempo y cualquier valor que queramos posteriormente y en tiempo real.

Uno de los problemas en las librerías base de Java es que a la hora de reproducir un sonido de un instrumento tenemos que hacerlo de una manera no musical. Es decir, si queremos reproducir una nota C (Do), tenemos que tratarla como un mensaje *MIDI* y mandar la información de la nota. Esta información se divide en tres valores clave, el valor de la nota (en este caso 60), longitud de la nota y además su velocidad de ataque, es decir, la intensidad.

JFugue nos permite de una manera musical y con un lenguaje de texto, saltarnos este paso. Nosotros podemos, a través de su API, decir que tiene que reproducir la nota A4, la nota C5 o la que queramos sin tener que preocuparnos de los valores interiores. Además, nos permite poder acceder a la biblioteca de sonidos de una manera sencilla por lo que podemos seleccionar el instrumento que queramos que suene simplemente indicando "Piano" o "Flute". Por estos motivos se ha optado por utilizar esta herramienta la cual dispone de una inmensa cantidad de herramientas musicales que pueden ayudar a evolucionar la aplicación

en futuras iteraciones. Podemos reproducir fragmentos musicales completos, desarrollar acordes complejos e incluso programar improvisaciones a lo largo de un tema musical.

6.3 JavaFX y Scene Builder

Para que el usuario pueda interactuar de una manera sencilla con la aplicación deberemos de programar una interfaz gráfica en nuestra aplicación. Esta interfaz gráfica será la interfaz con la que el usuario pueda acceder a las diferentes partes del programa de una manera visual.

Para poder dotar a nuestra aplicación de una interfaz gráfica agradable para el usuario se ha decidido usar la librería de JavaFX. La otra alternativa analizada fue Swing. En Swing existen numerosos componentes que nos permiten realizar nuestras aplicaciones de una manera sencilla, pero en JavaFX existen algunos más que nos facilitan esta tarea. Por otro lado, la personalización en Swing es algo tedioso ya que no podemos realizar cambios a grandes escalas mientras que en JavaFX disponemos de la posibilidad de usar CSS para la personalización. Otro añadido para JavaFX es la existencia de la librería JavaFXPorts que nos permitiría en un futuro realizar un port a plataformas móviles (iOs y Android) sin tener que cambiar el código, aunque al estar actualmente controlada por Gluon tendremos que realizar un pago para poder utilizar todos los servicios que esta ofrece.

Scene Builder es una aplicación aparte que nos permite crear ficheros que contienen en formato XML los diferentes componentes de una escena de JavaFX. Cada escena no es más que una ventana de la aplicación. Con Scene Builder podemos crear de una manera visual la interfaz gráfica. Sólo tendremos que arrastrar los componentes dentro de nuestra escena y luego cargar el fichero generado por el programa desde JavaFX. Esto nos ayuda ya que podemos ver el aspecto final que tendrá la aplicación y no es necesario tener que instanciar toda la interfaz gráfica dentro del código de la aplicación.

6.4 abc4j

abc4j es la librería que nos permite poder dibujar de una manera sencilla los pentagramas y las notas de los ejercicios teóricos. De esta manera podemos mostrar visualmente lo que nos pide el ejercicio. Esta librería también dispone de un intérprete musical como JFugue pero no dispone de tantas herramientas ni de una versión estable final.

La única característica que se usa de esta librería es la de representar en un pentagrama las notas facilitadas del ejercicio que estamos ejecutando. Esta librería tiene la posibilidad de crear un componente directo usado con Swing, pero se ha optado por guardar este componente como una imagen png y después usarla dentro de nuestro proyecto JavaFX.

7 Decisiones de diseño

En este apartado nos centraremos en decisiones que se ha tomado a lo largo del desarrollo. Empezaremos con las decisiones de valores clave para el análisis del audio y a continuación nos centraremos en las decisiones de diseño de los ejercicios teóricos y vocales. Especificaremos cómo se ha diseñado el análisis de ondas y de cómo se obtienen las frecuencias fundamentales de las ondas. En los ejercicios explicaremos cómo se han diseñado cada uno de ellos y cómo son los niveles de dificultad de los ejercicios teóricos.

7.1 Análisis de audio

Para la captación de audio se ha decidido usar una frecuencia de muestreo de 22’05kHz por defecto. Esta frecuencia nos permite realizar un análisis ajustado para lo que nosotros deseamos. Para llegar a esta decisión nos ayudamos de esta tabla.

Frecuencia muestreo (kHz)	Frecuencia máxima (kHz)
8	3.6
11.025	5
22.05	10
32	14.5
44	20
48	21.8
64	29.1
88.2	40
96	43.6

Tabla 7-1: Frecuencia muestreo y frecuencia máxima obtenible

Como podemos observar en la tabla, la frecuencia de muestreo de 22’05kHz nos permite obtener una frecuencia máxima de 10kHz. Esta frecuencia de muestreo es la misma usada por la radio para la transmisión de audio.

El otro factor importante para tener en cuenta para estos algoritmos de análisis es la decisión de cuántos puntos queremos que analice el algoritmo. Para un análisis rápido y fiable hemos decidido utilizar 4096 puntos de referencia.

Uno de los problemas tanto de los análisis FFT como DCT es que en las frecuencias bajas no existe mucha precisión. La primera frecuencia analizada es 0 Hz y la siguiente vendrá determinada por el número de puntos del algoritmo y la frecuencia de muestreo del audio. Siendo F_s la frecuencia sampleo, N el número de puntos del algoritmo, la frecuencia número n se obtendrá a partir de la siguiente fórmula:

$$f = \frac{n * F_s}{N}$$

Esto se traduce en que, en frecuencias bajas, si la frecuencia de muestreo es alta y el número de puntos del algoritmo es pequeño habrá saltos muy grandes entre las frecuencias analizadas y no permitirá una afinación correcta ya que existen muchos saltos. Por otro lado, si elegimos una N mucho mayor la precisión en graves mejorará mucho, pero a cambio tendremos

también un detalle demasiado elevado en frecuencias muy altas y aumentará el tiempo necesario para poder analizar el audio completo. Observemos el siguiente caso:
La nota La de la quinta octava en un piano (A5) sabemos que su frecuencia es 1760 Hz. Por otro lado, sabemos que las notas superiores e inferiores son La sostenido (A#5) y La bemol (Ab5) y corresponden con las frecuencias 1661 Hz y 1864 Hz. No es necesario un salto muy pequeño ya que las frecuencias fundamentales se encuentran a mucha distancia y por lo tanto un N muy grande provoca un análisis innecesario de frecuencias que no nos interesan. En resumen:

	Frecuencias Bajas	Frecuencias Altas	Latencia
N pequeño	Poca precisión	Precisión correcta	Baja
N grande	Precisión correcta	Precisión demasiado elevada	Alta

Tabla 7-2: Tabla comparativa puntos análisis FFT

Uno de los trucos que utilizan ahora cualquier herramienta que necesita una latencia muy baja para la representación del espectrograma de audio es la utilización de multi-ventanas. Esto no es otra cosa que utilizar diferentes puntos de referencia según avanzamos en frecuencias. Para los sonidos graves se usa una N grande hasta cierta frecuencia y luego se usa otra N más pequeña para reducir la latencia para las frecuencias superiores.

Para poder realizar un análisis más cómodo y rápido esto lo vamos a evitar ya que además la librería que nos está ayudando a realizar los algoritmos de FFT y DCT (JTransforms) no dispone de opciones para poder configurar esto.

Como para el análisis tanto FFT como DCT necesitamos un número mínimo de muestras dentro de nuestro fragmento de audio a analizar, que coincide con el número de puntos a analizar, tenemos que tener cuidado al elegir ya que, si la frecuencia de muestreo es inferior al número de puntos que estamos analizando, tardaremos más de un segundo en capturar todas las muestras a analizar por el algoritmo lo cual se traduce en una latencia muy alta para mostrar y dar un resultado al usuario. Lo ideal siempre es poder dar una solución rápida al usuario para que pueda corregir rápidamente la afinación de la nota que nos está facilitando por lo cual siempre deberíamos tener una elección de puntos inferior a la frecuencia de muestreo.

Con estos datos se realiza entonces un análisis FFT o DCT según hayamos elegido. Para ayudarnos a visualizar esto nos apoyamos en la herramienta MATLAB que tiene definidas ya las funciones FFT y DCT y creamos muestras de audio con frecuencias ideales para comprobar el comportamiento en nuestra implementación.

Tras verificar el correcto funcionamiento pasamos al audio real. Como se ha explicado ya en el apartado 4.3 *(Armónicos)* los armónicos son generados siempre y dependiendo de la intensidad de cada uno podemos modificar el timbre de un sonido.

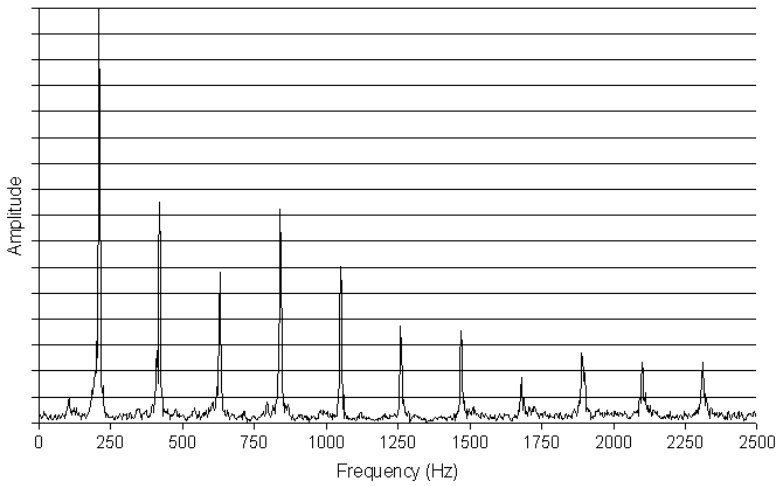


Figura 7-1: Chelo tocando un G# en la cuerda de D [11]

En la voz no es de otra manera y por ello para realizar una afinación mejor de la nota que estamos cantando debemos realizar también un análisis no solo de las frecuencias que existen en la muestra sino descartar frecuencias que podrían ser ruido con un análisis mediante los armónicos generados desde la frecuencia fundamental. Para ello se ha realizado un sistema de puntuaciones para cada frecuencia que se encuentra a partir de un umbral en la muestra analizada. Cada armónico que coincida para cada una de las frecuencias analizadas le otorga un punto y la que más puntuación obtenga será tratada como la frecuencia fundamental (o primer armónico) de la nota elegida. De esta manera nos aseguramos de que, aunque dado un timbre de voz más estridente que genere armónicos con un nivel más elevado que la frecuencia fundamental, obtengamos siempre la frecuencia fundamental y no posibles armónicos. Esto es importante realizarlo puesto que, si captásemos armónicos como notas fundamentales, nos alejaríamos mucho de la nota que el usuario está cantando en realidad. Para comprobar el correcto funcionamiento de este sistema de puntuación se realizó una prueba cantando la misma nota, pero una con un tono de voz apagado y otra con un tono de voz más estridente. En la parte superior de la gráfica podemos observar la frecuencia fundamental obtenida por el algoritmo y la nota correspondiente.

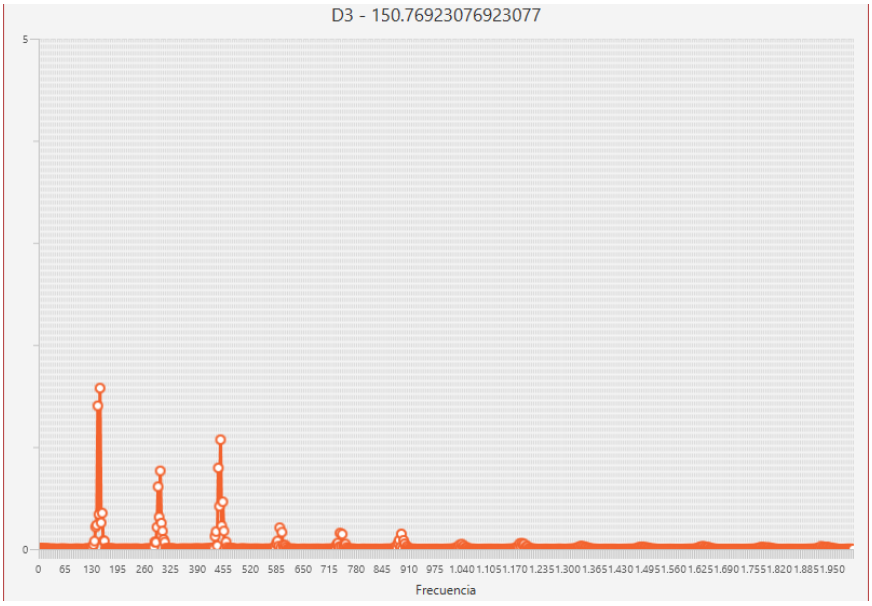


Figura 7-2: Espectro de frecuencias de nota apagada

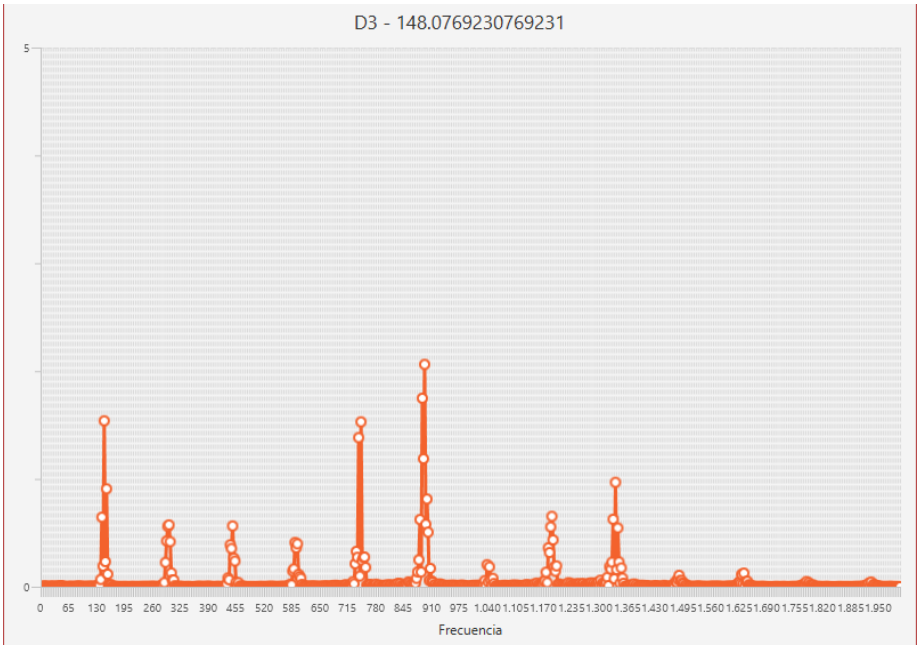


Figura 7-3: Espectro de frecuencias de nota estridente

En esta última gráfica observamos como el armónico que más presente está no es el mismo simplemente aumentando la estridencia con la que cantamos, pero la nota sigue siendo la misma. Dado que la puntuación máxima la obtiene el primer armónico gracias a este análisis, obtenemos siempre la nota correctamente. Esto también nos permite analizar correctamente sonidos que no sean sólo generados por la voz, también podemos analizar instrumentos musicales que tengan comportamientos diferentes en los armónicos.

El algoritmo FFT y DCT, como ya hemos visto anteriormente, nos genera un número de puntos de análisis que no podemos definir nosotros exactamente, es decir, no podemos realizar una medida exacta en la frecuencia 440 Hz ni en 1320 Hz, sino que el mismo algoritmo toma diferentes frecuencias que se generan según la iteración del algoritmo. Por este motivo no podemos obtener una afinación perfecta de la nota, pero sí muy aproximada debido a que se obtiene la afinación a través de los armónicos y las frecuencias analizadas más cercanas.

Imaginemos que realizamos sample con la frecuencia 440 Hz y sus correspondientes armónicos (880, 1320,1760,2200…). Una vez elegido el número de puntos para el algoritmo (FFT o DCT) observamos que no existen las frecuencias exactas sino una pequeña desviación de estas, es decir, un poco superiores o inferiores las más cercanas. Si quisiéramos aumentar la precisión para estas frecuencias tendríamos que aumentar el número de puntos que analiza el algoritmo lo cual supone aumentar el tiempo de procesado y aumenta la latencia considerablemente. Ya que esta precisión es bastante cercana a las frecuencias ideales no hace falta aumentar mucho el número de puntos que analizar. Con todo esto podemos obtener correctamente la nota que se está cantando o incluso tocando.

7.2 Ejercicios

Como hemos mencionado en la introducción, la idea de la aplicación es poder acercar más a los usuarios nuevos conocimientos de música o a aquellos que ya los conocen, reforzarlos de alguna manera. Por ello se han incluido dos bloques de ejercicios: unos teóricos y otros para la voz.

7.2.1 Ejercicios teóricos

Todos conocemos esa persona que sin saber conocimientos musicales puede tocar miles de canciones con la guitarra simplemente mirando los acordes en internet, pero esa persona realmente no sabe que está tocando ni por qué lo está tocando. La música actual parte de lo que se llama "armonía moderna". La base teórica de esta armonía es demasiado amplia como para poder plasmarla en una aplicación sencilla y que los usuarios no corran al verla. Por este motivo se ha hecho hincapié en la persona que toca esos acordes que no sabe qué notas son, pero tiene ganas de ampliar sus conocimientos.

El primer ejercicio propuesto es un ejercicio para reconocer notas. Una nota será reproducida y a continuación el usuario tendrá que elegir de entre unas cuantas posibles la respuesta que él considere correcta.

El siguiente paso es juntar dos notas y por eso se han realizado dos ejercicios destinado a esto: los intervalos. Un intervalo es la distancia entre dos notas ya sean en orden ascendente o descendente. Estos intervalos también tienen unos nombres específicos según la distancia entre las notas. Se han realizado dos ejercicios teóricos simples.
El primero de ellos es el reconocimiento de qué intervalo se trata. Con dos notas y un intervalo ascendente, se debe reconocer el intervalo que representan estas dos notas.
El segundo ejercicio consta de una nota y un intervalo, con ello deberemos saber qué nota es la correcta para completar el intervalo solicitado.

El siguiente paso ya es el de juntar tres o más notas. Esto son los acordes. Un acorde no es más que un conjunto de 3 o más notas tocadas a la vez. Estas notas no pueden ser aleatorias y se rigen según un orden y selección muy estricto. Este orden y selección es diferente para cada acorde y deberemos seleccionar de nuevo las notas para cada uno de ellos. Cómo son demasiados acordes los que existen, nos vamos a centrar en solamente dos para poder ver cómo funcionan exactamente.

Primero representemos la escala cromática de Do a Do (C a C), esto son todas las notas posibles.

C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C

Tabla 7-3: Notas de la escala cromática de Do (C)

Para un acorde mayor se tiene que seleccionar el primer, tercero y quinto grado y además deben existir entre ellos una distancia exacta.

Primer grado	4 semitonos	Tercer grado	3 semitonos	Quinto Grado

Tabla 7-4: Despliegue acorde mayor

Esto quiere decir que para el acorde mayor de C seleccionaremos las notas de C, E, G y para el acorde de E mayor usaremos las notas E, G#, B.

Para los acordes menores tendremos que usar otras notas diferentes. En este caso al ser menor indica que el tercer grado está disminuido y eso significa que deberemos quitarle un semitono.

Primer grado	3 semitonos	Tercer grado	4 semitonos	Quinto Grado

Tabla 7-5: Despliegue acorde menor

Ahora las notas de C menor serán C Eb G y las notas de E menor serán E G B.

Existen muchos más acordes, pero no son necesarios explicarlos todos y cada uno de ellos *(ver A.B*)
Al igual que para los intervalos, también se han realizado dos ejercicios. El primero de ellos en que se dispondrán de todas las notas que forman el acorde y a continuación se deberá elegir el acorde exacto que representa esas notas. El segundo se mostrará un acorde y todas las notas menos una que deberá averiguar el usuario.

Estos son los ejercicios teóricos implementados que ayudarán al usuario a entender un poco más de teoría musical. Se han implementado además tres niveles de dificultad para los ejercicios. Esto permite acercarnos poco a poco a la aplicación y no llevarnos un susto de repente con un ejercicio de un acorde enorme sin haber visto los acordes más sencillos.

	Fácil	Normal	Difícil
Reconocer nota	Solo disponibles notas sencillas	Disponibles todas las notas naturales, con bemoles y sostenidos	Disponibles todas las notas naturales, con bemoles y sostenidos
Rellenar Intervalo	Disponibles intervalos con notas naturales	Disponibles intervalos con todas las notas	Disponibles intervalos con todas las notas
Reconocer Intervalo	Disponibles intervalos con notas naturales	Disponibles intervalos con todas las notas	Disponibles intervalos con todas las notas
Rellenar Acorde	Solo disponibles notas naturales y acordes de hasta 3 notas	Disponibles todas las notas con acordes de hasta 4 notas	Disponibles todas las notas y todos los acordes. Además, no pregunta la nota fundamental
Reconocer Acorde	Solo disponibles notas naturales y acordes de hasta 3 notas	Disponibles todas las notas con acordes de hasta 4 notas	Disponibles todas las notas y todos los acordes

Tabla 7-6: Niveles de dificultad en ejercicios teóricos

7.2.2 Ejercicios de voz

Para los ejercicios de sonido hemos elegido 4 tipos diferentes:

- Ejercicio de afinación
- Ejercicio de afinación de intervalo con referencia
- Ejercicio de intervalos
- Ejercicio de potencia

El primer tipo es el ejercicio más sencillo y a la vez más útil de entre los 4 en el que tendremos que cantar una nota que nos dispondrán en pantalla. Con ayuda de una interfaz gráfica que reaccionará según cantamos, podremos afinar hasta conseguir la nota deseada que queríamos cantar para que el usuario, según avance en el ejercicio sea capaz de cantar a la primera la nota indicada y no tener que ir probando hasta conseguirla.

En el segundo ejercicio propuesto se proporciona un intervalo y puedes reproducir la primera nota de ese intervalo. El usuario debe cantar la segunda nota del intervalo para así poder afianzar los intervalos correctamente para el tercer ejercicio.

En el tercer ejercicio se proporciona un intervalo y el usuario debe cantar correctamente las dos notas del intervalo. Hasta que no se completa la primera nota, no se podrá continuar a la segunda nota de ese intervalo. Con ayuda del primer y segundo ejercicio, el usuario debería poder realizar este ejercicio con mucha facilidad.

El último ejercicio es más de potencia de voz. El usuario debe cantar o aguantar en silencio repetidas veces un tiempo definido aleatoriamente para cada ejercicio. El ejercicio termina correctamente cuando se han completado todas estas etapas. Si el usuario termina de cantar antes de tiempo, el ejercicio finalizará incorrectamente.

8 Desarrollo

En este apartado analizaremos cómo se ha estructurado internamente el programa. Dado que todo el programa está realizado con Java comentaremos parte de los diagramas de clases para poder obtener una mejor visión del proyecto y de cómo el lenguaje orientado a objeto nos facilita la programación para esta aplicación. Nos centraremos en dos aspectos importantes, las clases de audio, de ejercicios y de interfaz gráfica.

8.1 Clases AudioController y Spectrum

Existen dos clases que trabajan juntas y se encargan de todo el audio de la aplicación. En primer lugar, existe la clase *AudioController* que se encarga de obtener el audio del micrófono y aplicar el formato correspondiente para luego poder trabajar sobre él. Aparte esta clase tiene todos los datos necesarios para después un análisis correcto de los algoritmos que vamos a usar dentro de la clase *Spectrum*.

AudioController tiene formato de *singleton*, es decir, existe únicamente una instancia para la clase. Esto nos permite poder obtener audio desde otras clases de una manera sencilla sin tener que cerrar y abrir numerosas veces el micrófono lo cual provocaría lentitud en el proceso.

Spectrum será la clase encargada de analizar en su totalidad el audio. Podremos obtener tanto el nivel de audio del bloque de audio analizado como la descomposición de frecuencias a través de los algoritmos FFT o DCT.

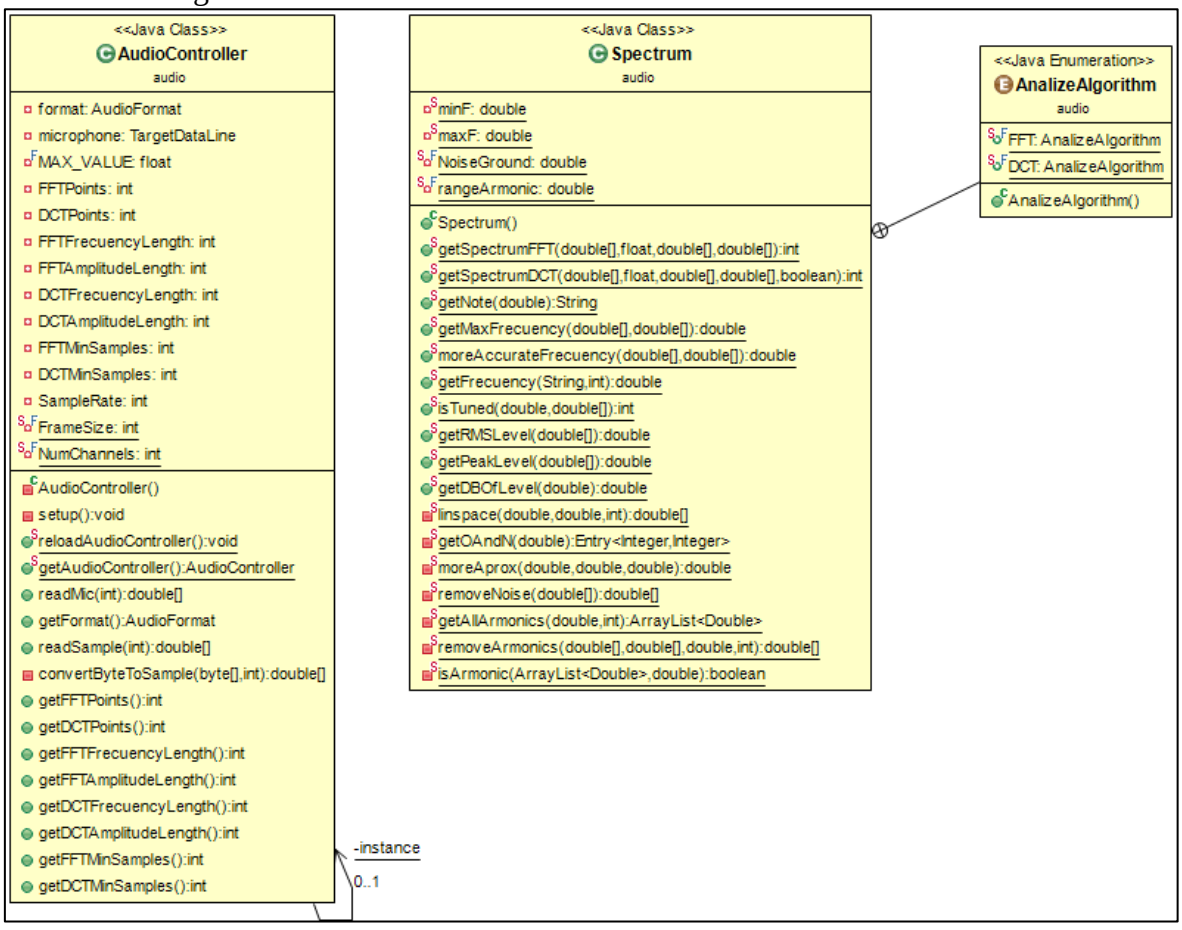
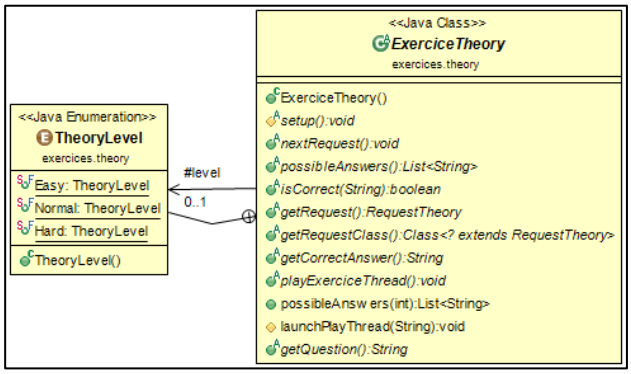


Figura 8-1: Clases AudioController y Spectrum

8.2 Ejercicios teóricos

Para los ejercicios teóricos se ha diseñado una clase abstracta que contendrá los métodos generales para todos los ejercicios. Esto nos permite que, tras una interfaz gráfica general, el comportamiento sea igual en todos los ejercicios y no sea necesario crear una ventana nueva para cada tipo de ejercicio. Tendremos los métodos generales de acceso a los ejercicios y cada una de las subclases, que serán los ejercicios, nos aportarán un comportamiento. La clase abstracta de la que hablamos es *ExerciceTheory*.



La clase tiene los métodos generales que nos permiten generar una nueva pregunta, obtener una lista de posibles respuestas, reproducir la pregunta y verificar si es correcta o no una respuesta. Todos estos métodos serán implementados en cada uno de los ejercicios teóricos que queramos hacer. Al disponer de métodos tan genéricos, es posible generar otros ejercicios en futuras iteraciones y escalar la aplicación.

Figura 8-2: Clase ExerciceTheory

Aparte de la clase abstracta *ExerciceTheory*, como hemos mencionado anteriormente, disponemos de una clase para cada tipo de ejercicio. Estos ejercicios además cuentan con otra clase interna que serán las respuestas. Estas respuestas serán subclases de otra clase abstracta llamada *RequestTheory*. Esta clase es la encargada de obtener las preguntas y los datos que se aportan para la resolución del ejercicio.

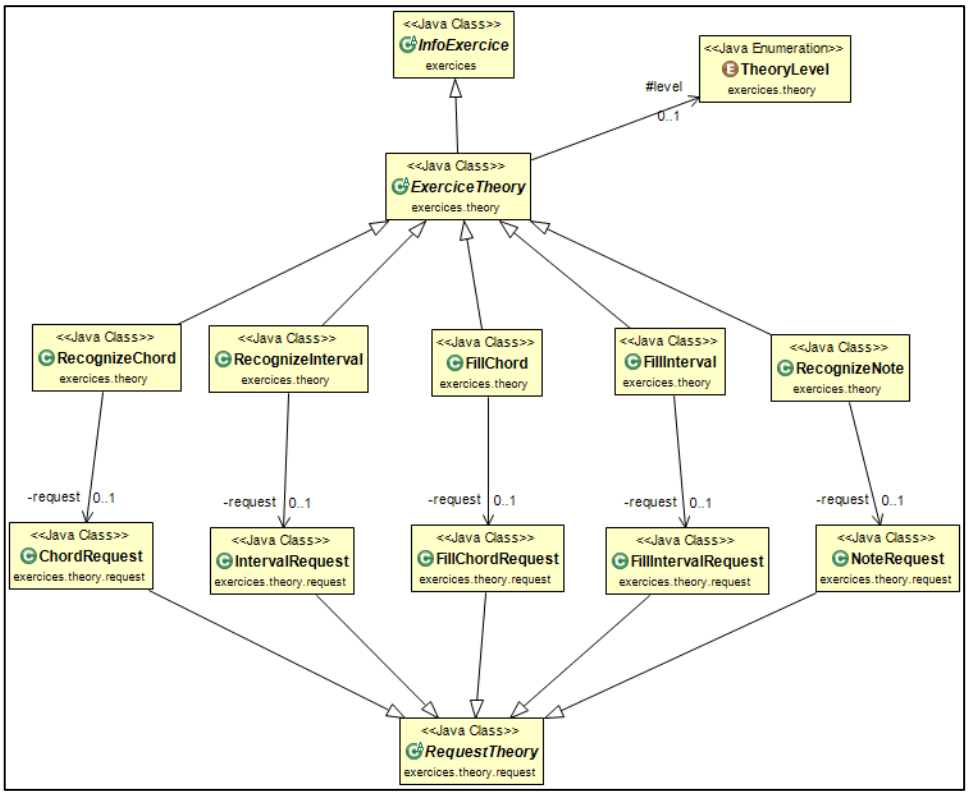


Figura 8-3: Clases de ejercicios teóricos

8.3 Ejercicios vocales

Para los ejercicios vocales se ha creado una clase que nos ayuda mucho con el audio. Esta clase se llama *Block*. Es la encargada de recoger el audio suficiente para el análisis de audio y realiza un análisis dependiente del tipo de *Block* que estemos creando. Podemos crear 3 tipos diferentes:

- **None:** Al crear este tipo de bloque, no se realiza un análisis de frecuencias. Si el bloque analizado no supera el umbral del micrófono de silencio, se cambiará el tipo a *Silence.*
- **Silence:** Este tipo de bloque es parecido al anterior, pero, aunque supere el umbral del micrófono, no se realizará el análisis de frecuencias.
- **Note:** Si elegimos este tipo de bloque, realizaremos un análisis de frecuencias con el algoritmo pertinente (FFT o DCT) siempre que supere el umbral del silencio del micrófono. Si no se supera, no será posible el análisis.

Estos tres tipos de *Block* nos permiten aumentar la facilidad a la hora de utilizar el audio dentro de las clases destinadas a los ejercicios vocales.

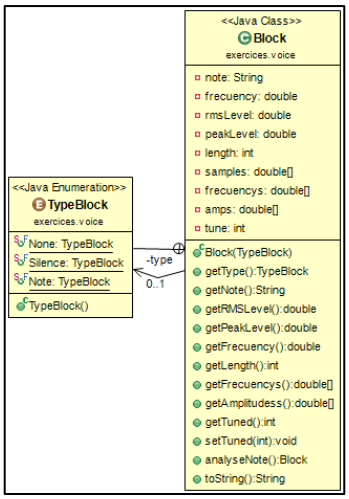


Figura 8-4: Clase Block

Al igual que en los ejercicios teóricos, en los ejercicios vocales también hemos decidido crear una clase abstracta con los métodos necesarios para la codificación de todos los ejercicios. La clase se llama *Exercice* y trabaja junto a una interfaz *ExerciceObserver*.

Exercice tiene una estructura muy definida. Tenemos métodos para comprobar si el ejercicio se ha terminado, si está correcto, si tiene algún audio que reproducir (como en el caso del ejercicio que hay que reconocer una nota), un método para generar una nueva pregunta y lo más importante un método de ejecución. La clase *Exercice* implementa la interfaz *Runnable*, esto quiere decir que es usada para la ejecución en hilos. Esto es debido a que el método de ejecución es apropiativo, y hasta que no finalice la ejecución no saldrá del método. Gracias a la interfaz *Runnable* y a la interfaz propia *ExericiceObserver*, podremos ejecutar en un hilo diferente a la interfaz gráfica el ejercicio y obtener un seguimiento.

ExerciceObserver nos permite mantener ese seguimiento. Esta interfaz dispone de cinco métodos diferentes que será invocados desde el ejercicio en el momento oportuno.

- ***startExercice***: Este método se invoca cuando el ejercicio ha comenzado. Esto suele ser cuando el micrófono ha superado el umbral de silencio.
- ***newBlockProcessed***: Cada vez que un *Block* sea analizado se invocará este método que nos aportará la información necesaria para ver si el ejercicio va en un camino correcto o no.
- ***nextStage***: Si el ejercicio tiene varias etapas, cuando alcancemos una etapa nueva se invocará el método.
- ***finishBlockProcessed***: Este indicará el final del ejercicio. También nos aportará información de si el ejercicio se ha ejecutado correctamente o no.
- ***errorFinish***: Solamente indicará un error en el caso de que la ejecución haya finalizado y no se ha realizado correctamente.

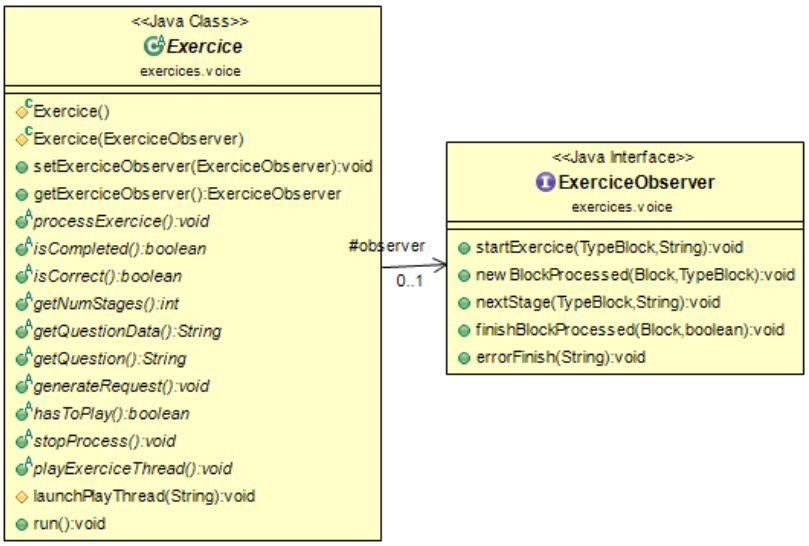


Figura 8-5: Clase Exercice e Interfaz ExerciceObserver

A continuación, mostraremos un pequeño diagrama de cómo quedan las clases orientadas a los ejercicios vocales. Como podemos ver, hay un ejercicio (*VoiceTune)* que hereda de otro ejercicio (*VoiceTuneReference)*, esto es debido a que el comportamiento de todos los métodos es prácticamente igual y la ejecución es la misma. El único detalle que cambia es que *VoiceTuneReference* sí que lleva una reproducción de audio, que sería la nota de referencia del intervalo. Esto nos permite poder reutilizar código de una manera rápida y sencilla.

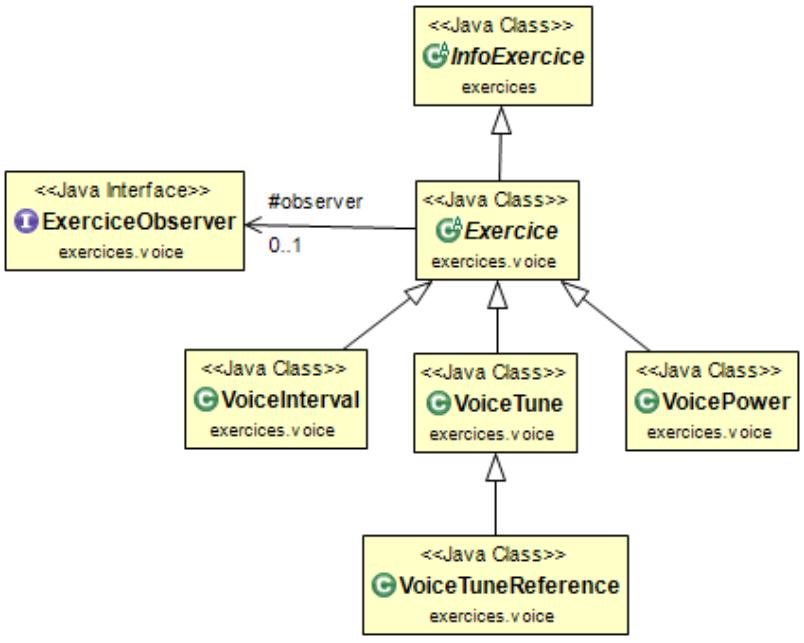


Figura 8-6: Clases de ejercicios vocales

8.4 Interfaz gráfica

Para la interfaz gráfica se han diseñado tres escenas principales. Estas escenas son las dedicadas a las tareas del menú principal, ejercicio teórico y ejercicio vocal. Al disponer de un diseño de clases abstractas podemos usar la misma escena para todos los ejercicios de cada tipo y así aumentar la versatilidad de la aplicación. Para el comportamiento de los

ejercicios teóricos solo disponemos de una clase la cual controla todo mientras que para los ejercicios vocales disponemos de dos clases que controlan la misma escena. Como hemos mencionaremos más adelante (*9.1 Ejercicios Vocales*), la escena de los ejercicios vocales tiene en su parte inferior un canvas (o lienzo) en el cual se dibuja, según avance el ejercicio, un afinador o una guía del ejercicio que nos permite seguir en tiempo real el avance del ejercicio. Para este comportamiento se ha diseñado una clase especial llamada *CanvasVoice* la cual implementa la interfaz *ExerciceObserver* y nos permite obtener todos esos eventos a lo largo de la ejecución del ejercicio. A parte de las escenas mencionadas tenemos otra que nos permite configurar diferentes aspectos de la aplicación como pueden ser los algoritmos de resolución o el nivel de los ejercicios.

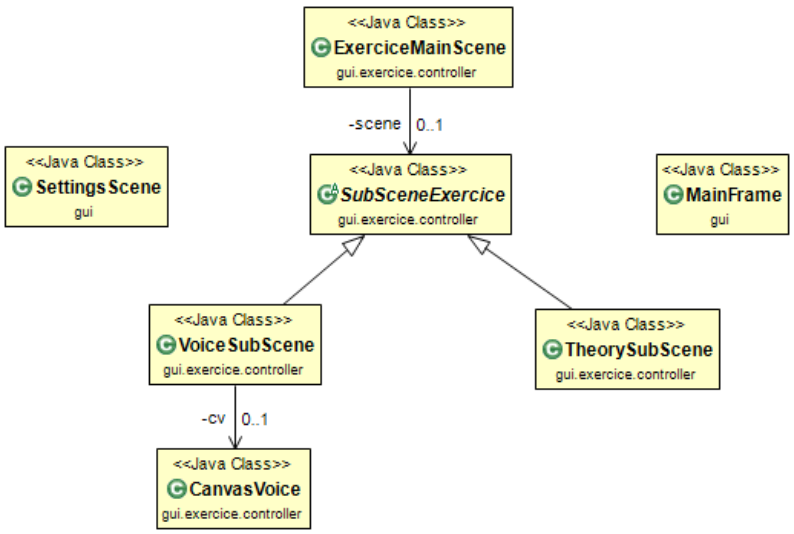


Figura 8-7: Clases de la interfaz gráfica

En el siguiente apartado veremos más en detalle como estas clases dan lugar a una interfaz gráfica sencilla e intuitiva para los usuarios. Explicaremos además como funcionan todos los botones que podamos encontrar a lo largo de la aplicación y cómo se resuelven los ejercicios correctamente.

9 Diseño

Como ya hemos mencionado anteriormente, toda la interfaz gráfica de la aplicación se ha realizado con JavaFX. A continuación, se incluyen unas capturas de la interfaz. En este apartado comentaremos las diferentes ventanas ya implementadas y mostraremos cómo funciona la aplicación.

Al iniciar la aplicación, lo primero que observaremos será un menú inicial con este aspecto:

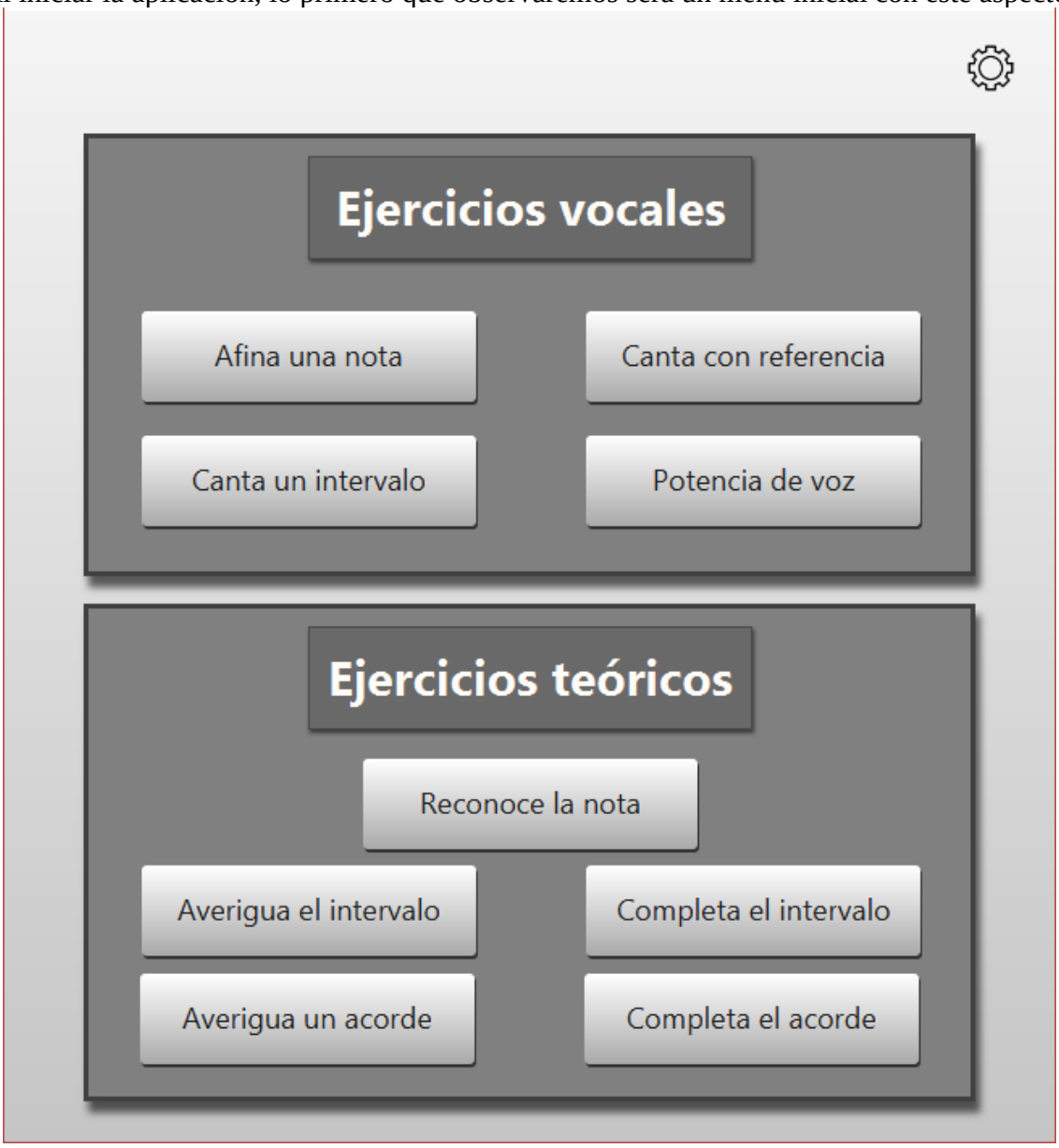


Figura 9-1: Menú principal

Como podemos observar, la aplicación está dividida en los dos grandes bloques que hemos mencionado anteriormente, por un lado, los ejercicios vocales y por otro lado los ejercicios teóricos. Además, disponemos de un menú de configuración que podemos acceder pulsado el botón situado en la esquina superior derecha.

9.1 Ejercicios Vocales

Todos los ejercicios vocales disponen de dos zonas. En la zona superior se indica un nivel de volumen del micrófono y las instrucciones del ejercicio. En la zona inferior se dispone de una guía visual que nos ayudará a obtener el resultado correcto de los ejercicios.
En el caso de afinación se dibujará un recuadro de diferentes colores (rojo, marrón y verde) y con un tamaño y posición concreta. En la zona superior también disponemos de un botón para generar otra pregunta o para volver al menú de inicio.

A continuación, se mostrarán las diferentes situaciones que pueden ocurrir en el caso de la afinación:

- **Sonido bajo:** La nota fundamental detectada es más grave que la esperada. *(véase Figura 9-2*: Afinador (sonido bajo)*)*
- **Sonido alto:** La nota fundamental detectada es más aguda que la esperada. *(véase Figura 9-3: Afinador (sonido alto))*
- **Sonido ajustado-alto:** La nota fundamental detectada está cerca, pero sigue siendo más aguda que la esperada. *(véase Figura 9-4: Afinador (sonido ajustado-alto))*
- **Sonido ajustado-bajo:** La nota fundamental detectada está cerca, pero sigue siendo más grave que la esperada. *(véase Figura 9-5: Afinador (sonido ajustado-bajo))*
- **Sonido ajustado:** La nota fundamental detectada está afinada correctamente. *(véase Figura 9-6: Afinador (sonido ajustado))*

Figura 9-2: Afinador (sonido bajo)

Figura 9-3: Afinador (sonido alto)

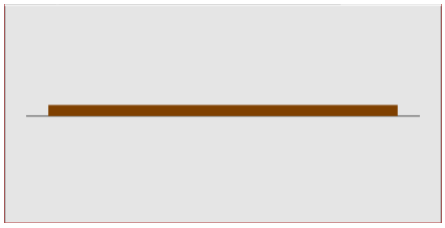


Figura 9-4: Afinador (sonido ajustado-alto)

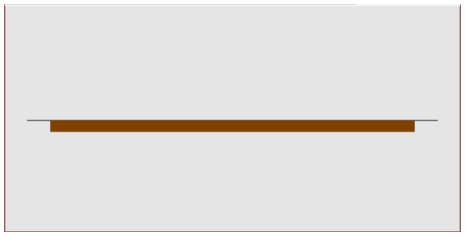


Figura 9-5: Afinador (sonido ajustado-bajo)

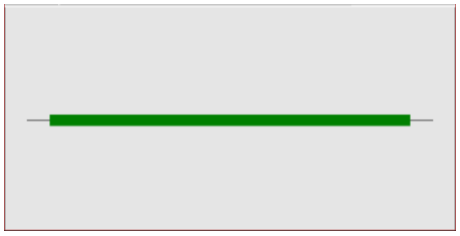


Figura 9-6: Afinador (sonido ajustado)

9.1.1 Afina una Nota

Al acceder al ejercicio se obtiene una interfaz sencilla con los datos claros sobre la nota que tenemos que intentar afinar.

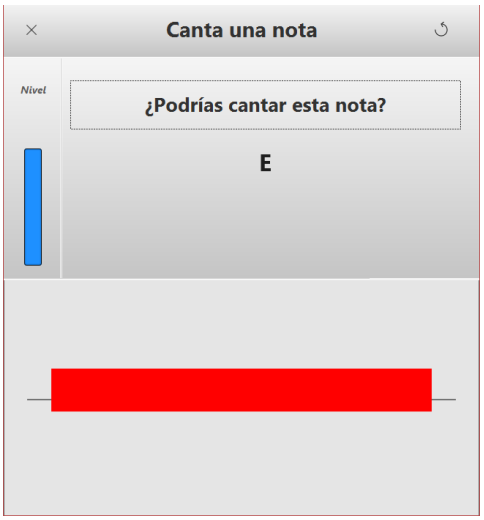


Figura 9-7: Ejercicio afinación (inicio)

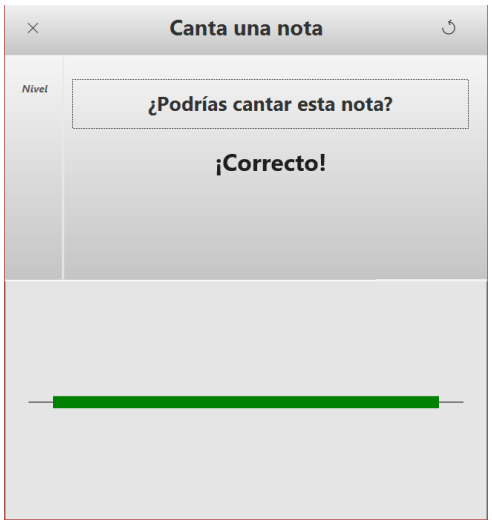


Figura 9-8: Ejercicio afinación (final)

En las figuras podemos observar una ejecución completa del ejercicio en la que nos pide cantar la nota E (Mi).

9.1.2 Canta con referencia

En este ejercicio se facilita un botón para poder reproducir la nota que nos facilitan como referencia en el intervalo.

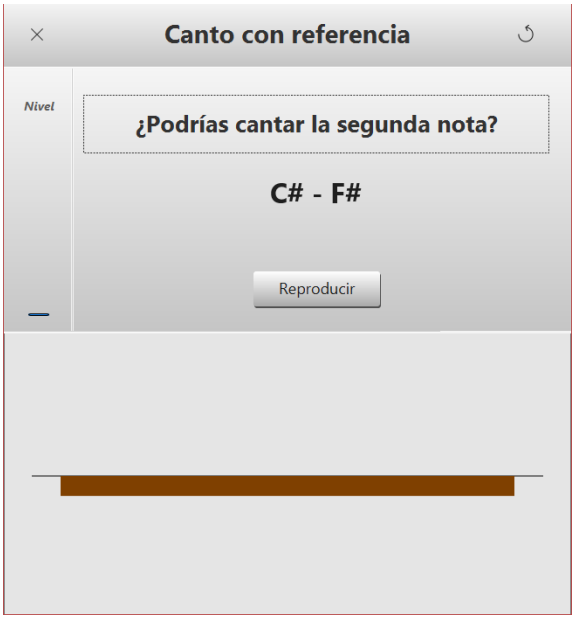


Figura 9-9: Canta con referencia

9.1.3 Canta un intervalo

En este ejercicio el afinador inferior se divide en dos zonas. Cada una de ellas servirá para cada una de las notas que tenemos que cantar en el ejercicio.

Figura 9-10: Canta un intervalo

9.1.4 Potencia de voz

En este ejercicio, la zona inferior no será un afinador sino una ayuda visual en la que cada zona de las que aparece dividido será el número de etapas del ejercicio (cantar, callar, cantar, callar, …) y se irán rellenando de color verde si la potencia es la adecuada y de color rojo si la potencia es inferior a la esperada o si toca callarse.

Figura 9-11: Potencia de voz

9.2 Ejercicios teóricos

En los ejercicios teóricos, disponemos de una apariencia similar a la anterior. En la parte superior disponemos de los datos relativos a la pregunta y en la parte inferior las posibles respuestas. Disponemos de dos modos para contestar a las preguntas que podremos configurar desde el menú de opciones. Si en opciones tenemos activada la opción "Drag Notes", para dar una solución tendremos que seleccionar la opción y arrastrarla al cuadro de la parte superior para verificarla. Si no tenemos activada esa opción simplemente tendremos que pulsar sobre la respuesta que queramos verificar. Si la opción es incorrecta, en el recuadro aparecerá una X y se podrá de color rojo. Podremos seguir verificando hasta que encontremos la solución correcta. También disponemos, como anteriormente, de un botón para generar otro ejercicio nuevo o para volver al menú principal.

9.2.1 Reconoce la nota

En este ejercicio podremos reproducir la nota y después verificar una respuesta que consideremos correcta. En este caso se mostrará el ejemplo de una ejecución completa, pero en posteriores ejercicios solo se mostrará el aspecto inicial.

Figura 9-12: Reconoce la nota (inicio)

Figura 9-13: Reconoce la nota (drag notes)

Figura 9-14: Reconoce la nota (fallo)

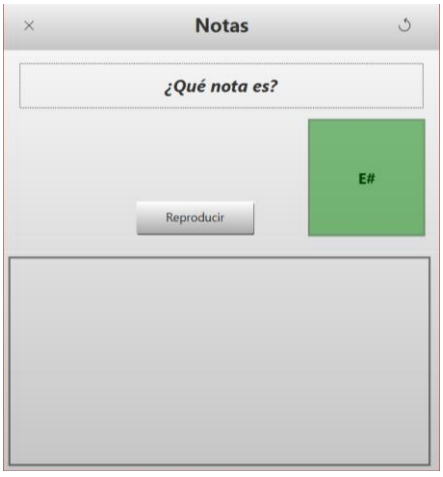


Figura 9-15: Reconoce la nota (correcto)

9.2.2 Averigua el intervalo

En este ejercicio disponemos también, a parte de la opción de reproducir el intervalo, de una ayuda visual con un pentagrama y las notas representadas.

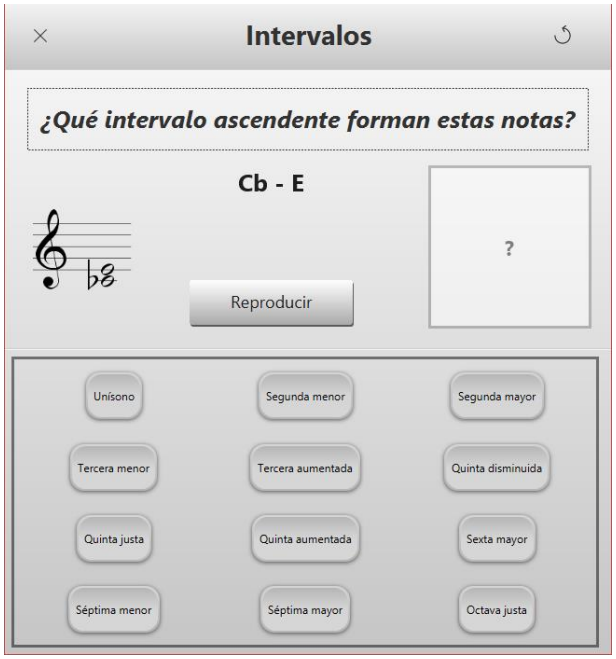


Figura 9-16: Averigua el intervalo

9.2.3 Completa el intervalo

Este ejercicio es similar al anterior donde debemos completar el intervalo propuesto.

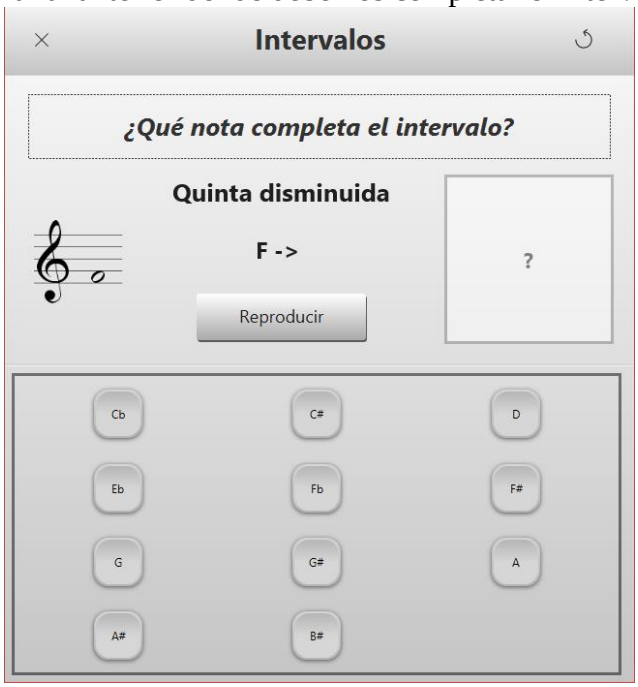


Figura 9-17: Completa el intervalo

9.2.4 Averigua un acorde

En este ejercicio, aparte de reproducir el acorde, si mantenemos el ratón encima del pentagrama, podremos obtener una ayuda que nos indicarán los grados que forman el acorde para poder completar correctamente el acorde

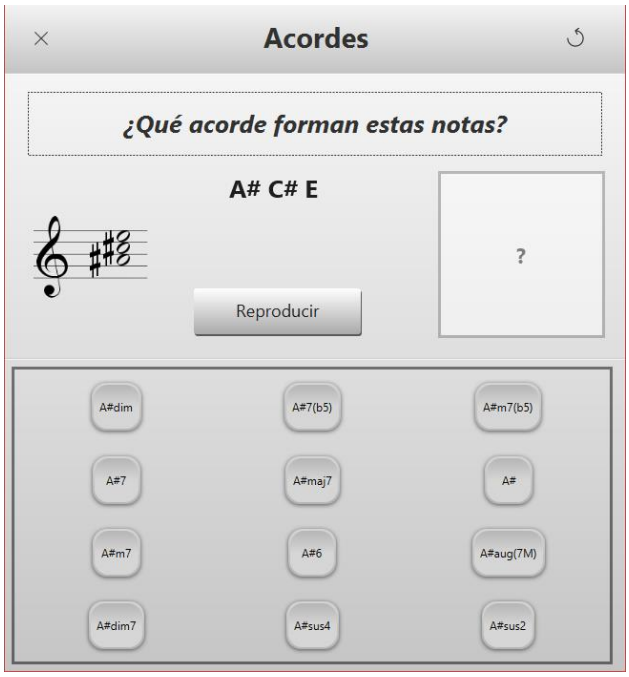


Figura 9-18: Averigua un acorde

9.2.5 Completa un acorde

Deberemos completar la nota que falta en el acorde con ayuda del audio y de las notas facilitadas y de la ayuda sugerida como en el ejercicio anterior.

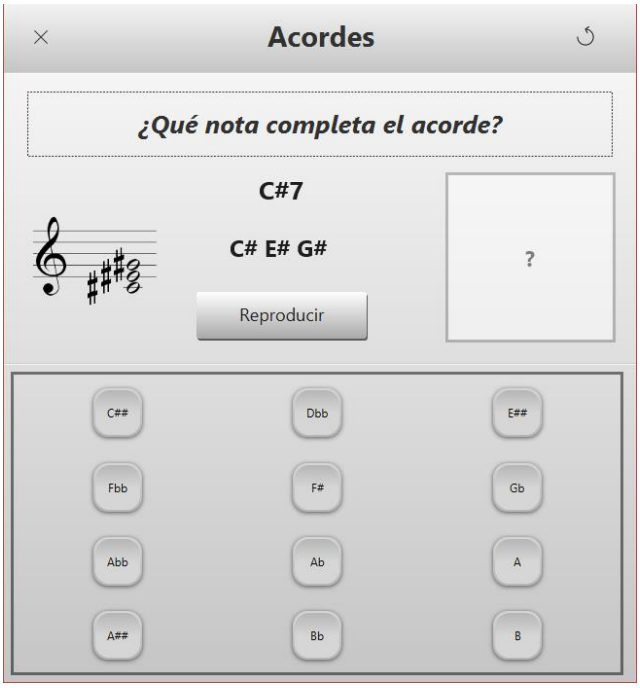


Figura 9-19: Completa un acorde

9.3 Menú de configuración

En este menú podremos configurar diferentes aspectos de la aplicación. Disponemos de 4 opciones generales y de 4 opciones avanzadas.

Ajustes generales:

- **Drag Notes:** Si se activa esta opción, las posibles respuestas generadas en los ejercicios teóricos aparecerán desordenadas a lo largo del tablero y deberemos arrastrarlas al cuadro de resolución para completar el ejercicio.
- **Umbral de micrófono:** Este *slide* nos ayudará para poder eliminar ruidos del micrófono. Según el micrófono obtenga un nivel de entrada de audio, se iluminará en verde a lo largo del *slide*. Deberemos mantener el nivel del *slide* en un punto localizado entre el ruido de ambiente y cuando cantamos de manera que sólo cuando cantemos se active el micrófono.
- **Nivel de ejercicios:** Con esta opción podemos modificar el nivel de los ejercicios teóricos para adaptarse a un nivel de usuario.
- **Máximas respuestas teóricas:** Podremos elegir el número máximo de posibles respuestas que se generan en los ejercicios teóricos.

Ajustes avanzados:

- **Algoritmo de análisis:** Podremos seleccionar el algoritmo que usará el programa para analizar el audio.
- **Frecuencia de sampleo:** Con esta opción elegiremos la frecuencia de captura del micrófono. No todas son compatibles con cualquier micrófono por lo que puede no funcionar en algunos dispositivos.
- **Tamaño FFT:** Aquí se elige el número de puntos que analizará el algoritmo de FFT
- **Tamaño DCT:** Al igual que la opción anterior modificaremos los puntos analizados en DCT.

Figura 9-20: Menú de configuración

10 Conclusiones y trabajo futuro

10.1 Conclusiones

Tras haber realizado este proyecto se ha obtenido un amplio conocimiento sobre el tratamiento de audio y cómo realizar un análisis correcto para poder analizar las notas que está cantando el usuario. Todo este conocimiento nuevo adquirido abre las puertas ante nuevas posibilidades de herramientas para diferentes tipos de usuarios.

La inclusión de generación de ejercicios aleatorios nos permite poder expandir la aplicación de manera que podamos usarla indefinidas veces ya que no siempre tendremos que rellenar los mismos ejercicios. Esto es importante dado que no nos centramos solo en unos temas y unos ejercicios definidos, sino que podremos navegar en ejercicios con preguntas diferentes cuando lo deseemos.

Los ejercicios de voz nos permiten mejorar nuestra afinación, aunque no solo valen para la voz, sino que, al realizar un análisis de las frecuencias armónicas, podemos también utilizarlo para instrumentos. Esto es especialmente útil en instrumentos de cuerda, como el violín o el contrabajo, pero también instrumentos de viento, como la flauta o el trombón, que quieran afinar a la perfección.

10.2 Trabajo futuro

Dado el diseño de clases implementado en el desarrollo de la aplicación, la ampliación de nuevos ejercicios puede resultar un trabajo fácil. Con los métodos genéricos disponibles en las clases de los ejercicios podremos realizar ejercicios de todo tipo teóricos. Además, en el caso de los ejercicios vocales, gracias al análisis en bloques podremos también realizar ejercicios fácilmente de todo tipo. Se pueden mejorar los aspectos teóricos incluyendo ejercicios de escalas o de armaduras de los pentagramas para poder reconocer tonalidades o mejorar los aspectos vocales con ejercicios como el cantar una secuencia de notas afinando a la perfección o simplemente acompañado de un fragmento musical viendo la afinación de aquello que vamos cantando. Puesto que la aplicación dispone de librerías externas como JFugue, que incluyen infinidad de herramientas para poder realizar todo tipo de ejercicios diferentes, se pueden implementar un lote de ejercicios mucho más técnicos que apliquen todos los conceptos que actualmente se aprende con la aplicación.

Esta aplicación solo es la punta del iceberg, hay mucha música que aprender y enseñar.

Referencias

[1] Wikipedia, «SingStar (serie),» [En línea]. Available: https://es.wikipedia.org/wiki/SingStar_(serie).

[2] UltraStar España, «UltraStar,» [En línea]. Available: https://ultrastar-es.org/es.

[3] Music Theory, [En línea]. Available: https://www.musictheory.net/exercises.

[4] J. Sánchez, «El sonido,» El Físico Loco, Febrero 2013. [En línea]. Available: http://elfisicoloco.blogspot.com/2013/02/el-sonido.html.

[5] Environmental Assay Inc., «EMF (Electromagnetic Field) Irritants Harmonics,» [En línea]. Available: http://www.emfrelief.com/harmonics.html.

[6] Wikipedia, «Transformada rápida de Fourier,» [En línea]. Available: https://es.wikipedia.org/wiki/Transformada_r%C3%A1pida_de_Fourier.

[7] Wikipedia, «Cooley–Tukey FFT algorithm,» [En línea]. Available: https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm.

[8] Wikipedia, «Discrete Sine Transform,» [En línea]. Available: https://en.wikipedia.org/wiki/Discrete_sine_transform.

[9] Wikipedia, «Discrete cosine transform,» [En línea]. Available: https://en.wikipedia.org/wiki/Discrete_cosine_transform.

[10] Wikipedia, «Ventana (función),» [En línea]. Available: https://es.wikipedia.org/wiki/Ventana_(funci%C3%B3n).

[11] C. Bercheck, «Cello Tone Analysis,» Vobarian, 02 Febrero 2007. [En línea]. Available: http://www.vobarian.com/celloanly/index.html. [Último acceso: 19 Marzo 2018].

[12] M. Raja, «La música es bella,» Octubre 2009. [En línea]. Available: http://musica-bella.blogspot.com/2009/10/cualidades-del-sonido-de-la-m.html.

[13] J. Wolfe, «Note names, MIDI numbers and frequencies,» Music Acoustics, [En línea]. Available: https://newt.phys.unsw.edu.au/jw/notes.html.

[14] Wikipedia, «Armónico,» Wikipedia, 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Arm%C3%B3nico.

[15] «Armonización y construcción de acordes,» Armonía moderna, [En línea]. Available: http://www.armoniamoderna.com/index.php?content=index:content:MySQL1/armoniaindex:51. [Último acceso: 22 Marzo 2018].

[16] «Comprender FFTs y Funciones Ventana,» National Instruments, 30 Marzo 2017. [En línea]. Available: http://www.ni.com/white-paper/4844/es/. [Último acceso: 5 Marzo 2018].

[17] Wikipedia, «Intervalo(música),» [En línea]. Available: https://es.wikipedia.org/wiki/Intervalo_(m%C3%BAsica). [Último acceso: 23 Marzo 2018].

[18] J. P. C. y. J. G. Bert, «Afinador de instrumentos,» Universidad Nacional de Rosario, https://www.dsi.fceia.unr.edu.ar/images/downloads/InfoElectronica/informe_proyecto.pdf, 2016.

[19] D. D. Fede, «Transformada de Fourier en Java,» GitHub, 20 Febrero 2010. [En línea]. Available: https://github.com/Uriopass/audio-analysis/blob/master/src/com/badlogic/audio/analysis/FourierTransform.java. [Último acceso: 26 Febrero 2018].

[20] L. Finney, «WRITING MUSIC IN JAVA: TWO APPROACHES,» OCI, Enero 2008. [En línea]. Available: https://objectcomputing.com/resources/publications/sett/january-2008-writing-music-in-java-two-approaches/. [Último acceso: 26 Marzo 2018].

[21] D. Koelle, The Complete Guide to JFugue, Second Edition, 2016, p. 206.

[22] L. Laurent, «Transformada de Fourier para el funcionamiento de un afinador electrónico,» Universidad Nacional del Sur, Argentina, Bahía Blanca, 2014.

[23] V. G. Ruiz, «Un espectrógrafo de secuencias de audio,» 24 Junio 2014. [En línea]. Available: https://w3.ual.es/~vruiz/Docencia/Apuntes/Multimedia/Audio_spectrum_visualization/index.html. [Último acceso: 28 Febrero 2018].

[24] J. Sastrón, «Entendiendo conceptos básicos de los analizadores FFT,» Producciones El Sótano, 7 Mayo 2017. [En línea]. Available: https://www.produccioneselsotano.com/entendiendo-conceptos-basicos-de-los-analizadores-fft/. [Último acceso: 28 Febrero 2018].

[25] vic, «Frecuencia de las notas musicales,» La tecla de ESCAPE, 19 Agosto 2015. [En línea]. Available: http://latecladeescape.com/h/2015/08/frecuencia-de-las-notas-musicales. [Último acceso: 16 Marzo 2018].

[26] P. Wendykier, «JTransforms,» GitHub, 2007. [En línea]. Available: https://github.com/wendykierp/JTransforms. [Último acceso: 28 Febrero 2018].

Glosario

API	Application Programming Interface
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform
Sample	Muestra de audio de tamaño definido
Nota	Sonido producido por una frecuencia fundamental determinada.
Intervalo	Diferencia de altura entre dos notas musicales.
Acorde	Conjunto de tres o más notas musicales que suenan de manera simultánea.
Armónico	Frecuencia múltiplo de la frecuencia fundamental de la onda
Interfaz gráfica	Entorno visual donde se permite una comunicación entre el usuario y la aplicación.
Canvas	Lienzo donde se permiten dibujar diferentes figuras geométricas como si se tratase de una pizarra

Anexos

A Manual de ejecución

Para ejecutar la aplicación se dispone de un ejecutable en ".jar" que contiene todo lo necesario de la aplicación. Una vez ejecutado por primera vez, se creará un fichero "settings.dat" el cual no se debe modificar directamente. Todo lo disponible en ese fichero deberá ser modificado desde la aplicación y su menú de configuración.

B Generación de acordes

Para la generación de acordes se ha realizado una codificación de los acordes generales hasta las undécimas de las mismas. No se han añadido, 13, 15, 17, 19 y demás por ser acordes demasiado específicos y realmente no son necesarios.
Primero se listarán los posibles acordes y se generará una tabla con todos los acordes generados desde la aplicación.

Sufijo	**Notas** **(Intervalo sobre la fundamental)**
5	Quinta justa
M	Tercera mayor Quinta justa
m	Tercera menor Quinta justa
aug	Tercera mayor Quinta aumentada
dim	Tercera menor Quinta disminuida
sus4	Cuarta justa Quinta justa
sus2	Segunda mayor Quinta justa
7	Tercera mayor Quinta justa Séptima menor
maj7	Tercera mayor Quinta justa Séptima mayor
6	Tercera mayor Quinta justa Sexta mayor
m6	Tercera menor Quinta justa Sexta mayor
m7	Tercera menor Quinta justa Séptima menor
m7(b5)	Tercera menor Quinta disminuida Séptima menor
dim7	Tercera menor Quinta disminuida Séptima disminuida
add9	Tercera mayor Quinta justa Novena mayor
m(7M)	Tercera menor Quinta justa Séptima mayor

aug(7M)	Tercera mayor Quinta aumentada Séptima mayor
aug7	Tercera mayor Quinta aumentada Séptima menor
7(b5)	Tercera mayor Quinta disminuida Séptima menor
maj7(9)	Tercera mayor Quinta justa Séptima mayor Novena mayor
7(9)	Tercera mayor Quinta justa Séptima menor Novena mayor
m7(9)	Tercera menor Quinta justa Séptima menor Novena mayor
7(9#)	Tercera mayor Quinta justa Séptima menor Novena aumentada
7(b9)	Tercera mayor Quinta justa Séptima menor Novena menor
6(9)	Tercera mayor Quinta justa Sexta mayor
maj7(9)(11)	Tercera mayor Quinta justa Séptima mayor Novena mayor Oncena disminuida
maj7(9)(#11)	Tercera mayor Quinta justa Séptima mayor Novena mayor Oncena justa
m7(9)(11)	Tercera menor Quinta justa Séptima menor Novena mayor Oncena disminuida
7(9)(11)	Tercera mayor Quinta justa Séptima menor Novena mayor Oncena disminuida

Lista de acordes generados:

Acordes	Notas
C5	C G
C	C E G
Cm	C Eb G
Caug	C E G#
Cdim	C Eb Gb
Csus4	C F G
Csus2	C D G
C7	C E G Bb
Cmaj7	C E G B
C6	C E G A
Cm6	C Eb G A
Cm7	C Eb G Bb
Cm7(b5)	C Eb Gb Bb
Cdim7	C Eb Gb Bbb
Cadd9	C E G D
Cm(7M)	C Eb G B
Caug(7M)	C E G# B
Caug7	C E G# Bb
C7(b5)	C E Gb Bb
Cmaj7(9)	C E G B D
C7(9)	C E G Bb D
Cm7(9)	C Eb G Bb D
C7(#9)	C E G Bb D#
C7(b9)	C E G Bb Db
C6(9)	C E G A D
Cmaj7(9)(11)	C E G B D F
Cmaj7(9)(#11)	C E G B D F#
Cm7(9)(11)	C Eb G Bb D F
C7(9)(11)	C E G Bb D F
C#5	C# G#
C#	C# E# G#
C#m	C# E G#
C#aug	C# E# G##
C#dim	C# E G
C#sus4	C# F# G#
C#sus2	C# D# G#
C#7	C# E# G# B
C#maj7	C# E# G# B#
C#6	C# E# G# A#
C#m6	C# E G# A#
C#m7	C# E G# B
C#m7(b5)	C# E G B
C#dim7	C# E G Bb
C#add9	C# E# G# D#
C#m(7M)	C# E G# B#

C#aug(7M)	C# E# G## B#
C#aug7	C# E# G## B
C#7(b5)	C# E# G B
C#maj7(9)	C# E# G# B# D#
C#7(9)	C# E# G# B D#
C#m7(9)	C# E G# B D#
C#7(#9)	C# E# G# B D##
C#7(b9)	C# E# G# B D
C#6(9)	C# E# G# A# D#
C#maj7(9)(11)	C# E# G# B# D# F#
C#maj7(9)(#11)	C# E# G# B# D# F##
C#m7(9)(11)	C# E G# B D# F#
C#7(9)(11)	C# E# G# B D# F#
Db5	Db Ab
Db	Db F Ab
Dbm	Db Fb Ab
Dbaug	Db F A
Dbdim	Db Fb Abb
Dbsus4	Db Gb Ab
Dbsus2	Db Eb Ab
Db7	Db F Ab Cb
Dbmaj7	Db F Ab C
Db6	Db F Ab Bb
Dbm6	Db Fb Ab Bb
Dbm7	Db Fb Ab Cb
Dbm7(b5)	Db Fb Abb Cb
Dbdim7	Db Fb Abb Cbb
Dbadd9	Db F Ab Eb
Dbm(7M)	Db Fb Ab C
Dbaug(7M)	Db F A C
Dbaug7	Db F A Cb
Db7(b5)	Db F Abb Cb
Dbmaj7(9)	Db F Ab C Eb
Db7(9)	Db F Ab Cb Eb
Dbm7(9)	Db Fb Ab Cb Eb
Db7(#9)	Db F Ab Cb E
Db7(b9)	Db F Ab Cb Ebb
Db6(9)	Db F Ab Bb Eb
Dbmaj7(9)(11)	Db F Ab C Eb Gb
Dbmaj7(9)(#11)	Db F Ab C Eb G
Dbm7(9)(11)	Db Fb Ab Cb Eb Gb
Db7(9)(11)	Db F Ab Cb Eb Gb
D5	D A
D	D F# A
Dm	D F A
Daug	D F# A#
Ddim	D F Ab
Dsus4	D G A

Dsus2	D E A
D7	D F# A C
Dmaj7	D F# A C#
D6	D F# A B
Dm6	D F A B
Dm7	D F A C
Dm7(b5)	D F Ab C
Ddim7	D F Ab Cb
Dadd9	D F# A E
Dm(7M)	D F A C#
Daug(7M)	D F# A# C#
Daug7	D F# A# C
D7(b5)	D F# Ab C
Dmaj7(9)	D F# A C# E
D7(9)	D F# A C E
Dm7(9)	D F A C E
D7(#9)	D F# A C E#
D7(b9)	D F# A C Eb
D6(9)	D F# A B E
Dmaj7(9)(11)	D F# A C# E G
Dmaj7(9)(#11)	D F# A C# E G#
Dm7(9)(11)	D F A C E G
D7(9)(11)	D F# A C E G
D#5	D# A#
D#	D# F## A#
D#m	D# F# A#
D#aug	D# F## A##
D#dim	D# F# A
D#sus4	D# G# A#
D#sus2	D# E# A#
D#7	D# F## A# C#
D#maj7	D# F## A# C##
D#6	D# F## A# B#
D#m6	D# F# A# B#
D#m7	D# F# A# C#
D#m7(b5)	D# F# A C#
D#dim7	D# F# A C
D#add9	D# F## A# E#
D#m(7M)	D# F# A# C##
D#aug(7M)	D# F## A## C##
D#aug7	D# F## A## C#
D#7(b5)	D# F## A C#
D#maj7(9)	D# F## A# C## E#
D#7(9)	D# F## A# C# E#
D#m7(9)	D# F# A# C# E#
D#7(#9)	D# F## A# C# E##
D#7(b9)	D# F## A# C# E
D#6(9)	D# F## A# B# E#

D#maj7(9)(11)	D# F## A# C## E# G#
D#maj7(9)(#11)	D# F## A# C## E# G##
D#m7(9)(11)	D# F# A# C# E# G#
D#7(9)(11)	D# F## A# C# E# G#
Eb5	Eb Bb
Eb	Eb G Bb
Ebm	Eb Gb Bb
Ebaug	Eb G B
Ebdim	Eb Gb Bbb
Ebsus4	Eb Ab Bb
Ebsus2	Eb F Bb
Eb7	Eb G Bb Db
Ebmaj7	Eb G Bb D
Eb6	Eb G Bb C
Ebm6	Eb Gb Bb C
Ebm7	Eb Gb Bb Db
Ebm7(b5)	Eb Gb Bbb Db
Ebdim7	Eb Gb Bbb Dbb
Ebadd9	Eb G Bb F
Ebm(7M)	Eb Gb Bb D
Ebaug(7M)	Eb G B D
Ebaug7	Eb G B Db
Eb7(b5)	Eb G Bbb Db
Ebmaj7(9)	Eb G Bb D F
Eb7(9)	Eb G Bb Db F
Ebm7(9)	Eb Gb Bb Db F
Eb7(#9)	Eb G Bb Db F#
Eb7(b9)	Eb G Bb Db Fb
Eb6(9)	Eb G Bb C F
Ebmaj7(9)(11)	Eb G Bb D F Ab
Ebmaj7(9)(#11)	Eb G Bb D F A
Ebm7(9)(11)	Eb Gb Bb Db F Ab
Eb7(9)(11)	Eb G Bb Db F Ab
E5	E B
E	E G# B
Em	E G B
Eaug	E G# B#
Edim	E G Bb
Esus4	E A B
Esus2	E F# B
E7	E G# B D
Emaj7	E G# B D#
E6	E G# B C#
Em6	E G B C#
Em7	E G B D
Em7(b5)	E G Bb D
Edim7	E G Bb Db
Eadd9	E G# B F#

Em(7M)	E G B D#
Eaug(7M)	E G# B# D#
Eaug7	E G# B# D
E7(b5)	E G# Bb D
Emaj7(9)	E G# B D# F#
E7(9)	E G# B D F#
Em7(9)	E G B D F#
E7(#9)	E G# B D F##
E7(b9)	E G# B D F
E6(9)	E G# B C# F#
Emaj7(9)(11)	E G# B D# F# A
Emaj7(9)(#11)	E G# B D# F# A#
Em7(9)(11)	E G B D F# A
E7(9)(11)	E G# B D F# A
F5	F C
F	F A C
Fm	F Ab C
Faug	F A C#
Fdim	F Ab Cb
Fsus4	F Bb C
Fsus2	F G C
F7	F A C Eb
Fmaj7	F A C E
F6	F A C D
Fm6	F Ab C D
Fm7	F Ab C Eb
Fm7(b5)	F Ab Cb Eb
Fdim7	F Ab Cb Ebb
Fadd9	F A C G
Fm(7M)	F Ab C E
Faug(7M)	F A C# E
Faug7	F A C# Eb
F7(b5)	F A Cb Eb
Fmaj7(9)	F A C E G
F7(9)	F A C Eb G
Fm7(9)	F Ab C Eb G
F7(#9)	F A C Eb G#
F7(b9)	F A C Eb Gb
F6(9)	F A C D G
Fmaj7(9)(11)	F A C E G Bb
Fmaj7(9)(#11)	F A C E G B
Fm7(9)(11)	F Ab C Eb G Bb
F7(9)(11)	F A C Eb G Bb
F#5	F# C#
F#	F# A# C#
F#m	F# A C#
F#aug	F# A# C##
F#dim	F# A C

F#sus4	F# B C#
F#sus2	F# G# C#
F#7	F# A# C# E
F#maj7	F# A# C# E#
F#6	F# A# C# D#
F#m6	F# A C# D#
F#m7	F# A C# E
F#m7(b5)	F# A C E
F#dim7	F# A C Eb
F#add9	F# A# C# G#
F#m(7M)	F# A C# E#
F#aug(7M)	F# A# C## E#
F#aug7	F# A# C## E
F#7(b5)	F# A# C E
F#maj7(9)	F# A# C# E# G#
F#7(9)	F# A# C# E G#
F#m7(9)	F# A C# E G#
F#7(#9)	F# A# C# E G##
F#7(b9)	F# A# C# E G
F#6(9)	F# A# C# D# G#
F#maj7(9)(11)	F# A# C# E# G# B
F#maj7(9)(#11)	F# A# C# E# G# B#
F#m7(9)(11)	F# A C# E G# B
F#7(9)(11)	F# A# C# E G# B
Gb5	Gb Db
Gb	Gb Bb Db
Gbm	Gb Bbb Db
Gbaug	Gb Bb D
Gbdim	Gb Bbb Dbb
Gbsus4	Gb Cb Db
Gbsus2	Gb Ab Db
Gb7	Gb Bb Db Fb
Gbmaj7	Gb Bb Db F
Gb6	Gb Bb Db Eb
Gbm6	Gb Bbb Db Eb
Gbm7	Gb Bbb Db Fb
Gbm7(b5)	Gb Bbb Dbb Fb
Gbdim7	Gb Bbb Dbb Fbb
Gbadd9	Gb Bb Db Ab
Gbm(7M)	Gb Bbb Db F
Gbaug(7M)	Gb Bb D F
Gbaug7	Gb Bb D Fb
Gb7(b5)	Gb Bb Dbb Fb
Gbmaj7(9)	Gb Bb Db F Ab
Gb7(9)	Gb Bb Db Fb Ab
Gbm7(9)	Gb Bbb Db Fb Ab
Gb7(#9)	Gb Bb Db Fb A
Gb7(b9)	Gb Bb Db Fb Abb

Gb6(9)	Gb Bb Db Eb Ab
Gbmaj7(9)(11)	Gb Bb Db F Ab Cb
Gbmaj7(9)(#11)	Gb Bb Db F Ab C
Gbm7(9)(11)	Gb Bbb Db Fb Ab Cb
Gb7(9)(11)	Gb Bb Db Fb Ab Cb
G5	G D
G	G B D
Gm	G Bb D
Gaug	G B D#
Gdim	G Bb Db
Gsus4	G C D
Gsus2	G A D
G7	G B D F
Gmaj7	G B D F#
G6	G B D E
Gm6	G Bb D E
Gm7	G Bb D F
Gm7(b5)	G Bb Db F
Gdim7	G Bb Db Fb
Gadd9	G B D A
Gm(7M)	G Bb D F#
Gaug(7M)	G B D# F#
Gaug7	G B D# F
G7(b5)	G B Db F
Gmaj7(9)	G B D F# A
G7(9)	G B D F A
Gm7(9)	G Bb D F A
G7(#9)	G B D F A#
G7(b9)	G B D F Ab
G6(9)	G B D E A
Gmaj7(9)(11)	G B D F# A C
Gmaj7(9)(#11)	G B D F# A C#
Gm7(9)(11)	G Bb D F A C
G7(9)(11)	G B D F A C
G#5	G# D#
G#	G# B# D#
G#m	G# B D#
G#aug	G# B# D##
G#dim	G# B D
G#sus4	G# C# D#
G#sus2	G# A# D#
G#7	G# B# D# F#
G#maj7	G# B# D# F##
G#6	G# B# D# E#
G#m6	G# B D# E#
G#m7	G# B D# F#
G#m7(b5)	G# B D F#
G#dim7	G# B D F

G#add9	G# B# D# A#
G#m(7M)	G# B D# F##
G#aug(7M)	G# B# D## F##
G#aug7	G# B# D## F#
G#7(b5)	G# B# D F#
G#maj7(9)	G# B# D# F## A#
G#7(9)	G# B# D# F# A#
G#m7(9)	G# B D# F# A#
G#7(#9)	G# B# D# F# A##
G#7(b9)	G# B# D# F# A
G#6(9)	G# B# D# E# A#
G#maj7(9)(11)	G# B# D# F## A# C#
G#maj7(9)(#11)	G# B# D# F## A# C##
G#m7(9)(11)	G# B D# F# A# C#
G#7(9)(11)	G# B# D# F# A# C#
Ab5	Ab Eb
Ab	Ab C Eb
Abm	Ab Cb Eb
Abaug	Ab C E
Abdim	Ab Cb Ebb
Absus4	Ab Db Eb
Absus2	Ab Bb Eb
Ab7	Ab C Eb Gb
Abmaj7	Ab C Eb G
Ab6	Ab C Eb F
Abm6	Ab Cb Eb F
Abm7	Ab Cb Eb Gb
Abm7(b5)	Ab Cb Ebb Gb
Abdim7	Ab Cb Ebb Gbb
Abadd9	Ab C Eb Bb
Abm(7M)	Ab Cb Eb G
Abaug(7M)	Ab C E G
Abaug7	Ab C E Gb
Ab7(b5)	Ab C Ebb Gb
Abmaj7(9)	Ab C Eb G Bb
Ab7(9)	Ab C Eb Gb Bb
Abm7(9)	Ab Cb Eb Gb Bb
Ab7(#9)	Ab C Eb Gb B
Ab7(b9)	Ab C Eb Gb Bbb
Ab6(9)	Ab C Eb F Bb
Abmaj7(9)(11)	Ab C Eb G Bb Db
Abmaj7(9)(#11)	Ab C Eb G Bb D
Abm7(9)(11)	Ab Cb Eb Gb Bb Db
Ab7(9)(11)	Ab C Eb Gb Bb Db
A5	A E
A	A C# E
Am	A C E
Aaug	A C# E#

Adim	A C Eb
Asus4	A D E
Asus2	A B E
A7	A C# E G
Amaj7	A C# E G#
A6	A C# E F#
Am6	A C E F#
Am7	A C E G
Am7(b5)	A C Eb G
Adim7	A C Eb Gb
Aadd9	A C# E B
Am(7M)	A C E G#
Aaug(7M)	A C# E# G#
Aaug7	A C# E# G
A7(b5)	A C# Eb G
Amaj7(9)	A C# E G# B
A7(9)	A C# E G B
Am7(9)	A C E G B
A7(#9)	A C# E G B#
A7(b9)	A C# E G Bb
A6(9)	A C# E F# B
Amaj7(9)(11)	A C# E G# B D
Amaj7(9)(#11)	A C# E G# B D#
Am7(9)(11)	A C E G B D
A7(9)(11)	A C# E G B D
A#5	A# E#
A#	A# C## E#
A#m	A# C# E#
A#aug	A# C## E##
A#dim	A# C# E
A#sus4	A# D# E#
A#sus2	A# B# E#
A#7	A# C## E# G#
A#maj7	A# C## E# G##
A#6	A# C## E# F##
A#m6	A# C# E# F##
A#m7	A# C# E# G#
A#m7(b5)	A# C# E G#
A#dim7	A# C# E G
A#add9	A# C## E# B#
A#m(7M)	A# C# E# G##
A#aug(7M)	A# C## E## G##
A#aug7	A# C## E## G#
A#7(b5)	A# C## E G#
A#maj7(9)	A# C## E# G## B#
A#7(9)	A# C## E# G# B#
A#m7(9)	A# C# E# G# B#
A#7(#9)	A# C## E# G# B##

A#7(b9)	A# C## E# G# B
A#6(9)	A# C## E# F## B#
A#maj7(9)(11)	A# C## E# G## B# D#
A#maj7(9)(#11)	A# C## E# G## B# D##
A#m7(9)(11)	A# C# E# G# B# D#
A#7(9)(11)	A# C## E# G# B# D#
Bb5	Bb F
Bb	Bb D F
Bbm	Bb Db F
Bbaug	Bb D F#
Bbdim	Bb Db Fb
Bbsus4	Bb Eb F
Bbsus2	Bb C F
Bb7	Bb D F Ab
Bbmaj7	Bb D F A
Bb6	Bb D F G
Bbm6	Bb Db F G
Bbm7	Bb Db F Ab
Bbm7(b5)	Bb Db Fb Ab
Bbdim7	Bb Db Fb Abb
Bbadd9	Bb D F C
Bbm(7M)	Bb Db F A
Bbaug(7M)	Bb D F# A
Bbaug7	Bb D F# Ab
Bb7(b5)	Bb D Fb Ab
Bbmaj7(9)	Bb D F A C
Bb7(9)	Bb D F Ab C
Bbm7(9)	Bb Db F Ab C
Bb7(#9)	Bb D F Ab C#
Bb7(b9)	Bb D F Ab Cb
Bb6(9)	Bb D F G C
Bbmaj7(9)(11)	Bb D F A C Eb
Bbmaj7(9)(#11)	Bb D F A C E
Bbm7(9)(11)	Bb Db F Ab C Eb
Bb7(9)(11)	Bb D F Ab C Eb
B5	B F#
B	B D# F#
Bm	B D F#
Baug	B D# F##
Bdim	B D F
Bsus4	B E F#
Bsus2	B C# F#
B7	B D# F# A
Bmaj7	B D# F# A#
B6	B D# F# G#
Bm6	B D F# G#
Bm7	B D F# A
Bm7(b5)	B D F A

Bdim7	B D F Ab
Badd9	B D# F# C#
Bm(7M)	B D F# A#
Baug(7M)	B D# F## A#
Baug7	B D# F## A
B7(b5)	B D# F A
Bmaj7(9)	B D# F# A# C#
B7(9)	B D# F# A C#
Bm7(9)	B D F# A C#
B7(#9)	B D# F# A C##
B7(b9)	B D# F# A C
B6(9)	B D# F# G# C#
Bmaj7(9)(11)	B D# F# A# C# E
Bmaj7(9)(#11)	B D# F# A# C# E#
Bm7(9)(11)	B D F# A C# E
B7(9)(11)	B D# F# A C# E